ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Sistema de Control de Acceso Físico Basado en Mensajería Instantánea

(Physical Access Control Using Instant Messaging)

Para acceder al Titulo de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Sergio Castillo Antoñán

Julio - 2020

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Sergio Castillo Antoñán
Director del TFG: Jorge Lanza Calderón
Título: "Sistema de Control de Acceso Físico Basado en Mensajería Instantánea"
Title: "Physical Access Control Using Instant Messaging"

Presentado a examen el día: 29 de Julio de 2020

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Miguel Ángel Manzano Ansorena
Secretario (Apellidos, Nombre): Jorge Lanza Calderón
Vocal (Apellidos, Nombre): Luis Francisco Díez Fernández

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecimientos

Después de un largo período de constante trabajo, he finalizado este proyecto.

Ha sido un proceso de aprendizaje continuo, con tiempos más fáciles y otros más difíciles. Gracias a la perseverancia que he podido mantener, con un apoyo moral inmenso por parte de mi familia. Por eso me gustaría agradecerles y dedicarles este trabajo de fin de grado, que es el fruto de 4 años de carrera desembocados en un proyecto de ámbito real. Ha sido muy beneficioso a nivel personal haber contado con ese apoyo que me ha hecho poder seguir adelante.

Han sido unos años duros, una experiencia nueva y siempre he podido contar con todo el personal tanto docente como estudiantil. Ha sido un placer haber colaborado con ellos como alumno y compañero respectivamente y he podido desarrollarme tanto educativa como personalmente. En especial, quería agradecer la importancia que ha tenido Jorge, mi tutor del TFG, el cual me propuso el proyecto y nunca ha subestimado mis capacidades para afrontar este reto. Él ha sido el que me dio la idea y me orientó sobre como llevar a cabo este proyecto, incitado no sólo por realizar el TFG, sino por poder algún día sacarlo al mercado, y quien sabe, si poder hacerlo comercial.

Resumen

Este proyecto está destinado a mejorar la calidad de vida de las personas y así hacerla adecuada a sus necesidades. Se puede realizar todo el proceso sin la necesidad de instalar alguna aplicación (versión web), o usando una aplicación de mensajería instantánea muy popular (Telegram).

Consiste en dos partes:

- La primera, se basa en la creación de claves temporales para los usuarios propietarios (residentes) que lo requieran y el posible visionado de estas claves por parte de los usuarios que no sean propietarios (visitantes).
- La segunda trata de la introducción de esa clave generada, junto con un identificador en un teclado numérico (pinpad), donde posteriormente se validará y se procederá a acceder o denegar la entrada a personas ajenas al lugar.

Con esto, se consigue que cualquier persona no residente que quiera realizar una visita notificada, disponga de un acceso temporal mediante esta clave volátil, para una fecha y hora determinadas, generado por un usuario residente.

Abstract

This proyect is made for improve people's lifes and adapt it to their needs. You can take along the process without installing any other application. It would be done from a popular instant messaging application or from a web version of it.

It is represented into two parts:

- The first one, is based on the setting up of temporal keys. The owners could make a new one, so the visitors can check them.

- The second one, is about inserting that generated key into a pinpad. Then, the key would be tested out. The system will accept or deny the entrance of these visitors.

Following this simple steps, every foreign person who wants to make a notified visit can have a temporal access with this volatile key for a given date and time.

Índice general

Índice de figuras

Índice de tablas

Índice de códigos

Acrónimos

Acrónimo	Español	Inglés
AES	Estándar de Encriptación Avanzada	Advanced Encryption Standard
Amp	Amperios	Amperes
AC-DC	Corriente Alterna - Corriente Continua	Alternating Current - Direct Current
API	Interfaz de Programación de Aplicaciones	Application Programming Interface
AWS	Servicios Web de Amazon	Amazon Web Service
CET	Tiempo Central Europeo	Central European Time
E2E	Extremo a Extremo	End to End
EEPROM	Memoria de Solo Lectura Programable y Borrable Eléctricamente	Electrically Erasable Programmable Read-Only Memory
GMT	Hora Media de Greenwich	Greenwich Mean Time
GND	Tierra	Ground
GUI	Interfaz Gráfica de Usuario	Graphical User Interface
HMAC	Hash MAC	Hash MAC
HOTP	Contraseña de Un Solo Uso Basada en HMAC	HMAC-based One-time Password Algorithm
HTML	Lenguaje de Marcas de Hipertexto	HyperText Markup Language
HTTP	Protocolo de Transferencia de Hipertexto	Hypertext Transfer Protocol
HTTPS	Protocolo Seguro de Transferencia de Hipertexto	HyperText Transfer Protocol Secure
I2C (I^2C)	Ciruito Inter-Integrado	Inter Integrated Circuit
ICSP	Programación Serie en Circuito	In Chip Serial Programmer
IDE	Entorno de Desarrollo Integrado	Integrated Development Environment
IGE	Extensión Infinita Ilegible	infinite garble extension
IV	Vector de Inicialización	Initialization Vector
IVA	Impuesto al Valor Agregado	Value Added Tax
JSON	Notación de Objeto de JavaScript	JavaScript Object Notation
KB	Kilo Bytes	Kilo Bytes
kHz	Kilo Hercios	Kilo Hertzs
LCD	Pantalla de Cristal Líquido	Liquid-Crystal Display

MB	MegaBytes	MegaBytes
Mhz	Megahercio	Megahertz
MTProto	Protocolo de Transporte Móvil	Mobile Transport Protocol
OTP	Contraseña de un Solo Uso	One-Time Password
PIN	Número de Identificación Personal	Personal Identification Number
PKI	Infraestructura de Clave Pública	Public Key Infrastructure
PWM	Modulación por Ancho de Pulsos	Pulse Width Modulation
QR	Respuesta Rápida	Quick Response
R/W	Lectura / Escritura	Read / Write
RFC	Petición de Comentarios	Request For Comments
RFID	Identificación por Radiofrecuencia	Radio Frequency Identification
RS	Selección de Registro	Registry Selector
RTC	Reloj en Tiempo Real	Real Time Clock
SCL	Pin de Reloj	Serial Clock
SDA	Pin de Datos	Serial Data
SHA	Algoritmo de Hash Seguro	Secure Hash Algorithm
SIM	Módulo de Identificación de Abonado	Subscriber Identity Module
SMS	Servicio de Mensajes Cortos	Short Message Service
SO	Sistema Operativo	Operative System
SQL	Lenguaje de Consulta Estructurada	Structured Query Language
SRAM	Memoria Estática de Acceso Aleatorio	Static Random Access Memory
SSO	Inicio de Sesión Único	Single Sign On
TOTP	Contraseña de Un Solo Uso Basada en Tiempo	Time-based One-time Password algorithm
USB	Bus Universal en Serie	Universal Serial Bus
UTC	Tiempo Universal Coordinado	Universal Time Clock
V	Voltios	Volts
V_0:	Pin de Contraste	Contrast Pin
VoIP	Voz sobre IP	Voice over IP
VCC	Voltaje en Corriente Continua	Voltage Common Collector
VSC	Visual Studio Code	Visual Studio Code
XOR	Disyunción Exclusiva	Exclusive OR
mA	Miliamperios	Milliamps
μs	Microsegundos	Microseconds

Glosario

[1] **Objeto:** "Se trata de un ente abstracto usado en programación que permite separar los diferentes componentes de un programa, simplificando así su elaboración, depuración y posteriores mejoras." [19]

[2] **Ubuntu:** Es un Sistema Operativo (SO) muy utilizado y libre. Puede implementarse tanto en un ordenador físico como utilizarse a través de la nube en un escritorio remoto. Es una distribución de Linux basada en Debian [26]. Se sustenta a base de vender servicios vinculados al SO y ofrecer soporte técnico.

[3] **Open Source:** "El software open source es un código diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente." [23]

[4] **Workers:** Un Worker es un script que se ejecuta en segundo plano, sin afectar o bloquear el desempeño del programa principal. Posteriormente, a través de un 'listener' en el programa principal, se puede recoger lo que retorne, el programa asignado al worker.

[5] **Promise:** "El objeto Promise (Promesa) es usado para computaciones asíncronas. Una promesa representa un valor que puede estar disponible ahora, en el futuro, o nunca." [18]

[6] **Plugin:** "Un plugin es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software." [25]

[7] **Token:** Es la utilización de un elemento sin significado intrínseco para representar y referenciar otros aspectos y datos, los cuales son privados personales y sensibles.

[8] **Tiempo Unix:** Es una forma de representar el tiempo en segundos desde el 1 de enero de 1970 (formato UTC). [34]

[9] **Hash:** Es una función que toma como entrada un mensaje de longitud variable y produce una salida de longitud fija. Son comunes para asegurar integridad y autenticación de mensajes. [42]

[10] **JavaScript:** "(JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js, Apache CouchDB y Adobe Acrobat. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa." [17]

1 Introducción

La visión que se tiene actualmente sobre la seguridad es muy relevante en la vida cotidiana. Ante la inseguridad de poder determinar cuán seguras están sus propiedades y qué individuos pueden acceder a ellas, se plantea la necesidad de disponer de mecanismos que protejan la privacidad y garanticen la seguridad, tanto virtual como física.

1.1. Motivación

Actualmente, existen plataformas de alquiler temporal de residencias. Estas empresas ofrecen casas rurales, apartamentos o cualquier tipo de vivienda a aquellas personas que lo soliciten. Sin embargo, el traspaso de las llaves para acceder al lugar y la implantación de políticas de control de acceso no está optimizado. Usualmente se hace de forma totalmente presencial. La mayoría de los arrendadores que quieren automatizar el proceso optan por depositar las llaves en algún lugar oculto. En su defecto, se utilizan mecanismos de acceso por clave o código secreto. Sin embargo, ese código suele ser fijo y único, requiriendo de actuaciones in-situ para modificarlo.

Esta última operativa también se observa en las residencias o comunidades de vecinos, donde muchas viviendas comparten un único acceso protegido por una verja. Ésta se desbloquea con la introducción de un código numérico único y conocido por todos los vecinos de la comunidad. Si se desea que alguien externo tenga acceso a la propiedad, la única forma es cediéndole el código. De esta forma, esa persona puede acceder indefinidamente a la parcela, ya que ese código no cambia.

Por ello, se plantea una solución capaz de evitar todos estos problemas de seguridad empleando herramientas tecnológicas que la mayoría de las personas tienen y a las que están habituadas. Actualmente, un 95.6 % de los españoles, según una empresa alemana de estadística con gran reputación internacional Statista, dispone en el hogar de al menos un teléfono inteligente [1]. En casi la totalidad de los casos, esos teléfonos incluyen aplicaciones de mensajería instantánea como Whatsapp o Telegram [11][12]. La sencillez de uso de las mismas ha hecho que su adopción y uso sea prácticamente masivo, bien desde el propio móvil o a través sus respectivas plataformas web.

1.2. Objetivos

El objetivo general del proyecto es la realización de una solución de gestión y control de acceso basada en soluciones de mensajería instantánea públicamente disponibles.

Los principales objetivos que se plantean para el presente trabajo están orientados a solventar los inconvenientes en cuanto a seguridad, confianza, fiabilidad y usabilidad apuntados:

- Demostrar la facilidad y usabilidad de una plataforma de mensajería instantánea para otros usos.

- Generar seguridad y fiabilidad a la hora de dar acceso a otra persona a tu propiedad, garantizando su autenticidad y sin ceder nada de información o material privado.
- Desarrollar un sistema innovador mediante recursos actuales utilizados habitualmente, y que suponga un gasto ínfimo tanto en instalación como mantenimiento.
- Crear un sistema escalable, el cual pueda ser perfeccionado y mejorado según las necesidades que surjan.
- Establecer los medios para la gestión autónoma de un sistema de gestión de políticas de control de acceso.

Para generar la solución objeto de este proyecto se aplicarán a un entorno real los conceptos teórico-prácticos adquiridos a través de las asignaturas cursadas. En este sentido, a título personal, el trabajo se afronta también con los siguientes objetivos:

- Consolidar los conocimientos de programación.
- Aprender a usar un Interfaz de Programación de Aplicaciones (API) e implementar una que habilite el acceso remoto a información.
- Manejar un microcontrolador tanto en programación como sus interfaces de entrada/salida.
- Aprender a programar aplicaciones en el ecosistema de una plataforma de mensajería instantánea.
- Poner en práctica algoritmos y protocolos de seguridad y gestión de claves.

1.3. Estructura del trabajo

Este documento se ha dividido en 5 capítulos. Tras el resumen y esta pequeña introducción y exposición de los objetivos, se presenta el capítulo de *Estado del arte*, donde se definen las bases sobre las que se sustenta el proyecto y se explican los conocimientos que se han requerido, así como las tecnologías existentes que han intentado plantear una solución como la que se muestra en este proyecto.

Posteriormente en la sección 3 *Bot de mensajería instantánea*, se describe cómo se ha realizado la interacción con la aplicación de automatización, al igual que toda su tecnología implicada. En el capítulo 4 *Equipamiento de validación*, se desarrollan todos los mecanismos empleados para llevar a cabo la parte práctica del proyecto relacionados con la parte física de control de acceso. Por último, se resume el desarrollo del proyecto extrayendo unas conclusiones y las líneas futuras que puede tomar este proyecto.

2 Estado del arte

Este capítulo está destinado a realizar un estudio sobre las bases en que se sustenta el proyecto, conocer las diferentes tecnologías que se han utilizado, así como los elementos hardware empleados.

2.1. Sistemas de autenticación y control de acceso

Desde hace unos años, la forma en la que vivimos ha cambiado radicalmente. El avance de la tecnología nos ha ayudado a hacer nuestra vida más segura y automatizada, sobre todo en lo que al acceso y uso de bienes materiales se refiere. A muchas personas, sobre todo de una avanzada edad, un hábito realizado durante toda su vida de forma natural, les supone un cambio muy grande el hecho de que dejen de hacerlo porque se les presenta un mecanismo más simple, rápido y eficaz. Desde el punto de vista de seguridad, los datos y el control de los dispositivos se hace aplicando soluciones que incluyen grabación de vídeo con cámaras, geolocalización, control remoto a través de Internet, etc. Muchas de estas soluciones tecnológicas, capturan y gestionan datos sensibles. Esto puede provocar el rechazo por parte de muchos usuarios al desconocer si se comercializa con esa información privada. Generalmente, para vigilar un lugar, controlar su acceso, etc. siempre se han utilizado personas. Este método es de los más fiables, ya que un ser humano tiene consciencia propia y es capaz de juzgar por sí mismo si lo que se produce está bien o mal. Sin embargo, una persona no puede estar vigilando y controlando las 24 horas del día, los 365 días del año. Además, esa persona debe percibir un sueldo reglado por la entidad correspondiente.

Algunas personas deciden no contratar este tipo de servicios por lo costosos que son. Sin embargo, utilizan sistemas básicos de autenticación y control de acceso. Estos son, como se ha mencionado, sistemas de protección por llave maestra, donde todo el mundo utiliza una copia de una llave maestra que sirve para dar acceso al recinto, o un panel numérico situado a la entrada, que controla el acceso, en el cual se ha supuesto un código predeterminado que da acceso. En el mejor de los casos, estos códigos se individualizan a cada persona. No obstante, en ambas técnicas, se puede traspasar esa llave/código a una persona ajena, quien puede acceder indefinidamente al lugar.

2.1.1. Sistemas fuertes de autenticación

Para evitar las deficiencias identificadas en los casos anteriores empleando soluciones mecánicas y de bajo coste, surgen mecanismos de autenticación robustos. Los más utilizados son los siguientes:

- **Un identificador y una Contraseña de Un Solo Uso (OTP):** Suplen al par más utilizado identificador y contraseña. Le da una mayor seguridad al sistema, ya que las contraseñas utilizadas tienen una duración limitada temporalmente y sólo tienen un único uso.
- **Certificados digitales almacenados en dispositivos seguros:** Utiliza certificados X.509 que aplican una tecnología avanzada de codificación que permite calcular o firmar mensajes sin tener que compartir el secreto. El identificador es un certificado público que es firmado y en consecuencia garantizado por una autoridad de certificación reconocida. El usuario debe proporcionar un secreto para poder utilizar los distintos elementos criptográficos.
- **Tarjetas inteligentes:** Se almacena el identificador junto con la contraseña sobre una tarjeta inteligente. Sin la tarjeta, y sin su código Número de Identificación Personal (PIN), no se puede acceder a la contraseña.
- **Biometría:** La autenticación biométrica se basa en la verificación de un elemento del cuerpo del usuario (huella dactilar, iris, patrones faciales, etc.).

Estos son algunos de los mecanismos más utilizados actualmente para omitir la contratación de empresas o terceras personas para vigilar y garantizar la entrada verificada a un recinto de una forma automatizada. Hoy en día, incluso algunos de estos se consideran vulnerables, por lo que para aumentar su veracidad se implementa una autenticación multi-factor.

Por un lado se emplea algo que el usuario sabe o conoce, que puede ser un código secreto, una contraseña alfanumérica, etc. y forma parte de ese secreto que no se debe compartir. Por otro lado, lo que ese usuario posee. Esto engloba todo el material usable que generalmente tiene una sola persona, ya que son elementos individuales y no transferibles. Pueden ser tarjetas inteligentes, tokens USB, o cualquier dispositivo electrónico que almacene internamente información que se proporcione en un sistema de autenticación. Por último, lo que el usuario es. Esto está intrínsecamente relacionado con la biometría y todos sus mecanismos para su verificación. Por ejemplo, el uso de una huella dactilar o la retina ocular junto con el iris permite la validación de una parte corporal totalmente única de cualquier otro ser humano.

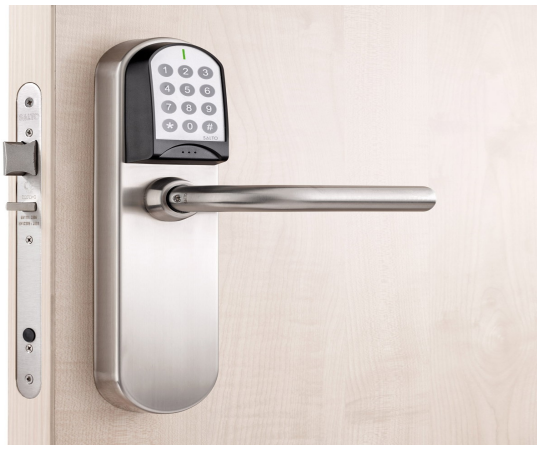
La afirmación de cuán seguro es un sistema y qué mecanismos ha de usar para garantizar completamente la entrada a un lugar protegido depende de cómo se desee proteger, de qué tecnología se va a emplear y qué se quiere proteger, es decir, su importancia.

La creación de todos estos métodos ha hecho que no se posicione uno como el método más adecuado, sino que se da la opción al instalador del sistema de que juzgue qué mecanismo es el que mejor se adapta a las necesidades del usuario y posteriormente sea el que emplee.

2.1.2. Sistemas de autenticación comerciales

Todos estos mecanismos teóricos mencionados anteriormente se han utilizado para crear sistemas robustos en cuanto a seguridad. Una empresa dedicada a este sector como es *Salto Systems* [2], tiene un catálogo con los siguientes productos:

- **Cerraduras electrónicas:** Las más sólidas en cuanto a seguridad se refiere implementan un doble factor. Requieren una credencial (como puede ser una tarjeta) y un código PIN. Además, están conectadas a Internet para monitorizar y controlar los accesos. El modelo XS4 E60K se puede apreciar en la figura 2.1a.
- **Tecnología de proximidad:** Estos sistemas implican el uso de una tarjeta, pulsera u otro componente con una banda magnética. El modelo utilizado generalmente para taquillas se puede apreciar en la figura 2.1b.
- **Lectores murales:** Son puntos de conectividad para la verificación de *Contactless Smart Cards* (tarjetas especialidades de uso por aproximación). La comprobación de éstas se realiza a través de la red (bluetooth generalmente). El modelo XS4 se puede apreciar en la figura 2.1c.



(a) Cerradura electrónica XS4 E60K

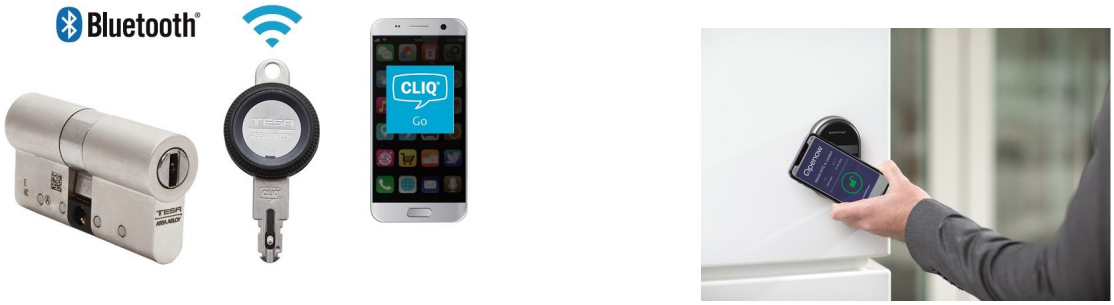
(b) Cerraduras para taquillas XS4

(c) Lector mural modular de proximidad RFID XS4

Figura 2.1: Productos comerciales de la empresa Salto Systems

Otras empresas del sector como *Tesa Assa Abloy* [3], tienen en su catálogo:

- **Cilindros electrónicos CLIQ GO:** Se utiliza una llave electrónica maestra para la apertura de puertas, donde se comunican con otros dispositivos a través de una aplicación móvil vía Bluetooth para traspasar datos de acceso. Se puede ver su forma en la figura 2.2a.
- **Lectores Identificación por Radiofrecuencia (RFID) y sistemas *SmartAir*:** Mediante el uso de un sensor con radiofrecuencia, un teléfono móvil o un dispositivo especializado, se puede interactuar con el receptor para validar las claves internas. Todo controlado bajo una aplicación que gestiona los accesos y monitoriza el estado de la cerradura. Su interacción se puede visualizar en la figura 2.2b.



(a) Cilindro electrónico junto con su llave electrónica y la aplicación controladora

(b) Interacción de un teléfono móvil con el sistema SmartAir

Figura 2.2: Productos comerciales de la empresa Tesa Assa Abloy

Al igual que estas empresas, muchas otras como *HID* [4], *FAC-Seguridad* [5], *CISA* [6], etc. se dedican a crear mecanismos para garantizar la seguridad en accesos mediante cerraduras, unos más complejos y robustos que otros pero todos con una característica repetida, el elevado precio tanto de adquisición del material como de la mano de obra necesaria para su instalación. Todo ello que hace que queden descartados estos sistemas si no se requiere una seguridad tan estricta.

2.2. Sistemas de mensajería

La mensajería ha ido evolucionando de forma muy rápida, sobre todo en las últimas décadas. Desde tiempos inmemoriales, se tenía la necesidad de emplear mensajería para transmitir información. Desde nuestros ancestros usando señales de humo, hasta los egipcios utilizando su famoso lenguaje de símbolos, etc. son ejemplos de ello. En el siglo XIX el telégrafo fue una revolución, que dio paso al teléfono a finales de siglo. Actualmente, la utilización de un teléfono móvil es algo cotidiano, pero en su día supuso una gran revolución en la en la que nos comunicarnos al permitir hacerlo de forma ubicua y en cualquier momento [27]. Lo normal es disponer de un teléfono inteligente, es decir, un dispositivo móvil con pantalla táctil que ofrece funciones

de ordenador, portable y con conexión a Internet.

Estamos habituados a la utilización de una aplicación de mensajería instantánea, la cual se comunica a través de Internet y puedes mandar mensajes de texto con contenido multimedia adjunto (audio, fotos o vídeo). La latencia de un mensaje desde que se envía hasta que es recibido es mínima. Inicialmente estas aplicaciones se basaron el uso del Servicio de Mensajes Cortos (SMS), lo que suponía un coste por cada mensaje enviado. Desde que se implementaron ciertas aplicaciones que permiten la mensajería gratuita, su uso se ha incrementado exponencialmente. Consecuentemente, las llamadas telefónicas, para lo que estaba provisto en un principio el móvil, se han visto lentamente relegadas, también afectadas por las conversaciones a través de Voz sobre IP (VoIP) o los sistemas push-to-talk que emplean las aplicaciones de mensajería .

En España la aplicación de mensajería instantánea por excelencia es Whatsapp, aunque estos últimos años, debido a fallos reiterados, Telegram está ganando bastante aceptación. Lo que más llama la atención sobre su principal competidor es "el gran valor añadido de la seguridad tanto en la encriptación de los mensajes, como en la posibilidad de la autodestrucción de los mismos y la posibilidad de la verificación en dos pasos. Además, la posibilidad de multidispositivo y de poder usarlo directamente con la aplicación de escritorio sin necesidad de tener un smartphone."[33]

2.2.1. Whatsapp

Esta aplicación de mensajería, como se ha comentado anteriormente es la más utilizada en España. Su lema sobre el que se basa la plataforma es "Mensajería confiable. Simple. Segura." El cual exponen en su página web principal [7].

Sus desarrolladores aseguran que la plataforma es rápida, simple y segura tanto en su mensajería instantánea como en sus llamadas. Esta sencillez en su interfaz y operabilidad es la que hace que haya ganado tantos adeptos a lo largo de estos años en los que se ha impuesto sobre sus competidores. Las personas, sobre todo de la tercera edad, buscan cosas sencillas con las que poder desenvolverse, ya que nunca han tenido que convivir con este tipo de tecnologías y el cambio puede resultar algo abrumador.

Esta plataforma cuenta con un tutorial de uso de 7 sencillos pasos o un vídeo de 2 minutos donde se explican con detalle las interacciones que debe seguir cualquier usuario que desee utilizar su aplicación [8].

En el tema de la seguridad, Whatsapp implementa el cifrado Extremo a Extremo (E2E) en todos sus chats incluyendo mensajes, fotos, videos, mensajes de voz, llamadas y documentos. Según los desarrolladores, solamente el emisor y receptor del contenido son capaces de descifrarlo en claro. Este proceso de encriptación se puede observar en detalle en la siguiente referencia [9].

2.2.2. Telegram

Es una de las aplicaciones de mensajería instantánea más utilizadas actualmente. Sus competidores directos son WhatsApp (que se ha detallado anteriormente), Line, WeChat, Facebook

messenger, SnapChat, Hangouts, Skype y Viber entre otros muchos.

La mayoría de las personas utilizan Whatsapp como aplicación de mensajería instantánea por defecto, a pesar de los problemas ya mencionados. Esto viene motivado por el efecto de seguimiento de tendencia. Una aplicación de mensajería tiene sentido si tus contactos la utilizan, pues de lo contrario no te puedes comunicar con nadie de tu circulo. Según muestran estadísticas del grupo alemán Statista, en el año 2016 un 97 % de la población española eran usuarios activos de esta plataforma [28]. Sin embargo, a nivel global estas estadísticas bajan, siendo la lista desarrollada en la figura 2.3.

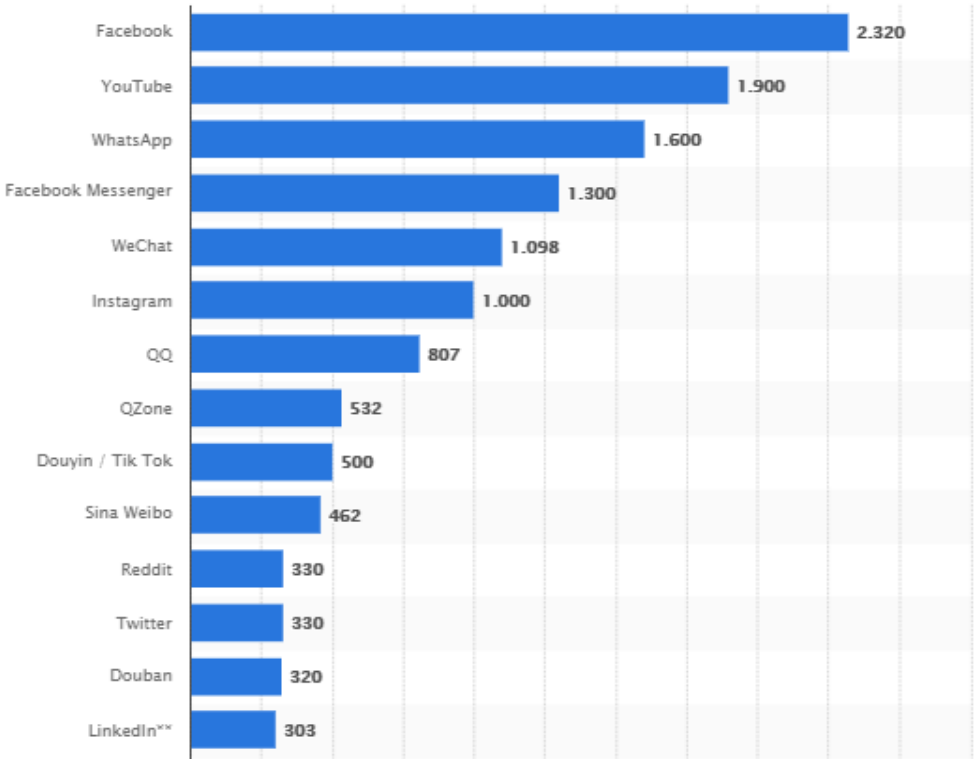


Figura 2.3: Ranking mundial de redes sociales por número de usuarios

Telegram fue desarrollado por los hermanos Nikolai y Pavel Durov, en Rusia en 2013. Es una aplicación que se enorgullece de seguridad y privacidad, lo cual prima mucho actualmente y hace que se posicione en lo alto, cuando se habla de estos temas. Sus múltiples funcionalidades se ven representadas en la figura 2.4.

Utiliza un mecanismo de seguridad conocido como Protocolo de transporte móvil (MTProto) [24] al contrario que su competidor directo en España; Whatsapp, que utiliza cifrado E2E. Está desarrollado bajo un estándar abierto, a base de API Java, aunque los servidores no son de código abierto. Actualmente, no se le han encontrado vulnerabilidades y se considera un cifrado seguro. Todas las conexiones se realizan a través del servidor de Telegram por lo que nunca va

a haber una conexión directa entre usuarios. Telegram además implementa ese cifrado E2E en una parte de su plataforma llamada *Chats Secretos*, para garantizar aún más la confidencialidad de los mensajes.

Figura 2.4: Funciones Telegram

Telegram expone este mecanismo usado en la comunicación, el cual está documentado y se sabe exactamente cuál es su funcionamiento. Sin embargo, otras compañías como Whatsapp ocultan toda su API y únicamente exhiben el algoritmo de cifrado que utilizan, sin saber cómo está implementado. El funcionamiento del algoritmo MTProto, el cual se puede observar en la imagen 2.5 es el siguiente:

Antes de que un mensaje (individual o grupal) se transmita a través de una red utilizando un protocolo de transporte, se cifra y se le agrega un encabezado externo que consiste en un identificador de clave de 64 bits, *auth_key_id* (identificando de forma exclusiva una clave de autorización para el servidor y el usuario) y una clave de mensaje de 128 bits, *msg_key*. La clave de autorización *auth_key* combinada con la clave de mensaje *msg_key* define una clave real de 256 bits, *aes_key* y un Vector de Inicialización (IV) de 256 bits, *aes_iv*, que se utilizan para cifrar el mensaje usando el cifrado Estándar de Encriptación Avanzada (AES)-256 en modo Extensión Infinita Ilegible (IGE). La parte inicial del mensaje a cifrar contiene datos variables (sesión, identificador del mensaje, número de secuencia, server salt, etc.) que influyen en la clave del mensaje (y, por lo tanto, en la clave AES e IV). En MTProto 2.0, la clave del mensaje se define como los 128 bits intermedios del Algoritmo de Hash Seguro (SHA)-256 del cuerpo del mensaje (incluyendo sesión, ID del mensaje, relleno, etc.) antepuestos por 32 bytes tomados de la clave

de autorización. En la versión 1.0 de MTProto, la clave del mensaje se calculaba como los 128 bits inferiores de SHA-1 del cuerpo del mensaje, excluyendo los bytes de relleno.

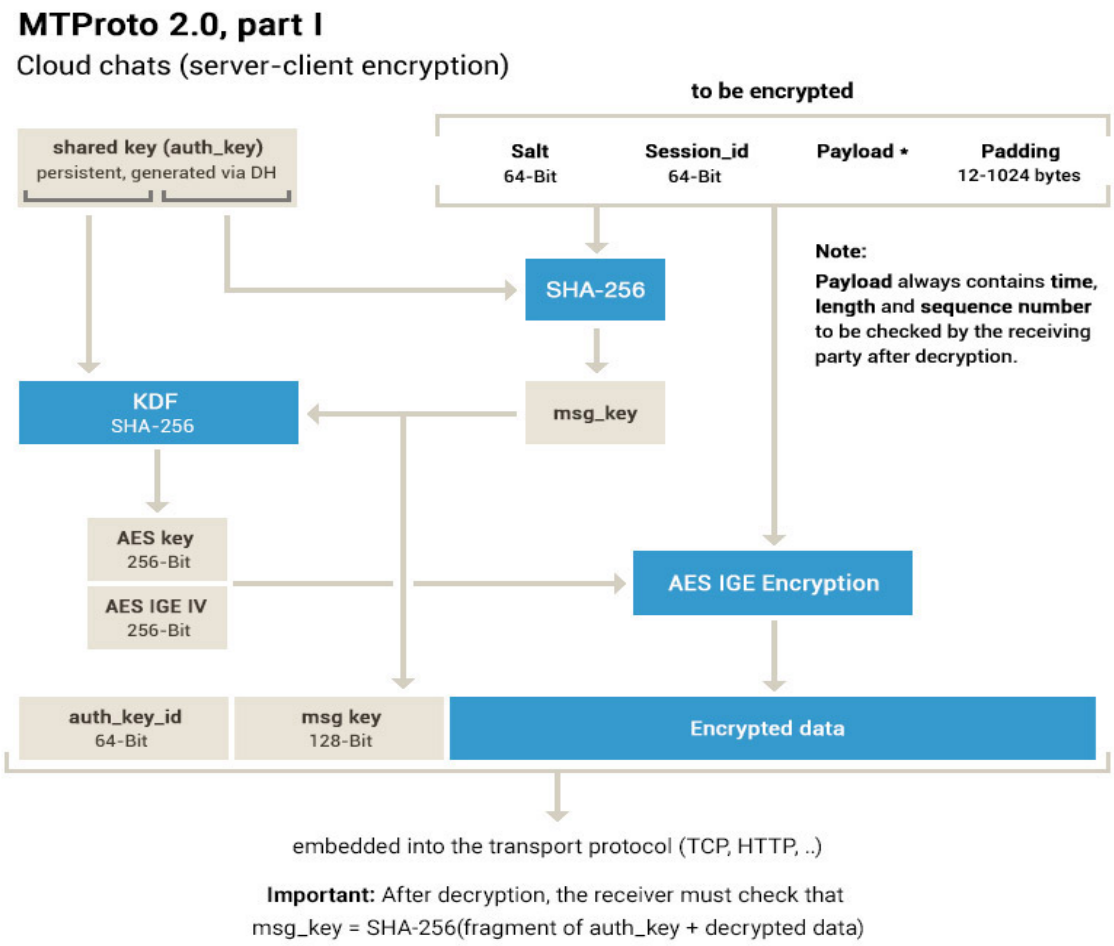


Figura 2.5: MTProto 2.0

Si se enfrentan ambas aplicaciones, Whatsapp domina en número de usuarios sobre Telegram. En términos de seguridad y privacidad, la balanza se inclina del lado de Telegram por los métodos utilizados. Otros aspectos relevantes para el usuario común, como son la personalización y la disponibilidad multiplataforma están implementados de forma más adecuada en Telegram. Así por ejemplo Telegram dispone de cliente autónomo en múltiples entornos, mientras que Whatsapp requiere siempre tener una sesión activa en el teléfono móvil para operar. Ambas plataformas incluyen stickers, emojis, gifs, archivos, o incluso la posibilidad de hacer llamadas.

A la hora de afrontar desarrollos externos sobre un sistema de mensajería, hay que considerar no solo el volumen de usuarios sino también la disponibilidad de opciones, librerías, etc. Este es el motivo por el que a lo largo de este TFG se emplea Telegram. Telegram, además de ser una

solución de código abierto, dispone de una API para la generación de bots. Un bot es un programa que permite automatizar el intercambio de mensajes y establecer conversaciones dinámicas y ajustadas a las necesidades del usuario en un momento dado.

Su facilidad en el manejo de la API para crear bots funcionales, programables y personalizados es el punto fuerte de Telegram. WhatsApp por el contrario, limita mucho el uso de bots, ya que con la nueva política de privacidad e intervención gubernamental, se ha limitado el reenvío de mensajes o el flood/spam (como han reflejado en su página web) [10], que no deja de ser para lo que está destinado un bot; el envío de mensajes programados, reenvío masivo de mensajes, etc. [30]. Además no tienen una API dedicada especialmente a la programación de los mismos y el funcionamiento es totalmente diferente. No podrían realizarse muchas funciones esenciales, sino que se deben situar en un dominio, junto con una tarjeta Módulo de Identificación de Abonado (SIM). Ahí se redirigirán todos los mensajes entrantes al servidor propio donde se analizarán y se crearán las respuestas determinadas.

2.2.2.1. Entorno gráfico

Telegram es una plataforma donde toda la información que se intercambia se almacena en la nube. Esto quiere decir que no depende de ningún dispositivo especial donde se redirijan todas las peticiones. Tiene un almacenamiento ilimitado, donde se guardan sin coste todos los historiales de conversación. Esto ofrece la posibilidad de interactuar con el bot desde cualquier plataforma con acceso a Internet. La forma más común de usar la plataforma Telegram es a través de su aplicación móvil, disponible tanto para Android [11], como para IOS [12] o Windows Phone [13], es decir, prácticamente la mayoría de los dispositivos móviles actuales. Ofrece también una versión para entornos de escritorio, *Telegram Desktop* [14]. No obstante, si se encuentra en otro dispositivo que no sea de estas características como puede ser un dispositivo con un software propio, que no esté vinculado a una tienda de aplicaciones, se puede usar su interfaz web [15] o crear una aplicación propia a través de la API adecuada.

No obstante, la versión accesible a través del navegador presenta limitaciones en cuanto al acceso a la cámara para tomar instantáneas, enviar una ubicación, contactos, programar mensajes, etc. Pero para realizar la interacción con un bot a través de un chat es totalmente compatible y funcional, sin ninguna restricción. La aplicación gráfica con la que va a tener que interactuar el usuario es la representada en la figura 2.6.

Además todas sus aplicaciones para los SO mencionados son de código libre, estando su código accesible a través de un repositorito abierto. En la siguiente referencia se puede acceder a cada implementación de código respecto de su plataforma [16].

2.2.2.2. Interfaz de programación

Dentro de la plataforma de Telegram se ofrecen 2 tipos de API diferentes. La primera permite crear una aplicación propia basada en los servidores de Telegram en la que se puede modificar la interfaz y adaptarla al gusto propio. De esta manera se puede generar una aplicación de Telegram similar a la oficial, pero con una interfaz más atractiva y cómoda para el usuario. Un

ejemplo de uso de esta API es PlusMessenger.

Figura 2.6: Aplicación de Telegram

La segunda API da la posibilidad de programar un bot funcional. Telegram ofrece una librería a través de la que generar programas que interactúen de forma autónoma directamente con los servidores, recibiendo, gestionando y respondiendo a los mensajes que se transmitan. Estos programas actúan como usuarios de Telegram, pero no hay ninguna persona tras ellos, sino un conjunto de reglas preestablecidas en el propio programa.

Para la creación y utilización de un bot no es necesario registrarlo como una cuenta nueva, lo que conllevaría un número de teléfono dedicado, sino que cualquier usuario puede generar uno. Al implementar el programa sobre la API se le provee de un token[7] y una dirección Protocolo Seguro de Transferencia de Hipertexto (HTTPS) con la que referenciarlo. No se puede proveer una dirección Protocolo de Transferencia de Hipertexto (HTTP) porque no se considera lo suficientemente seguro. Esa dirección es general para cualquier bot, pero el token es específico, sirviendo como identificador único.

La operativa de un bot comprende el procesado de todas las peticiones de los usuarios al bot. Existen 2 maneras de interpretarlas y atenderlas. La forma más común es usar los servidores de Telegram para recibir esas peticiones mediante polling, procesarlas y posteriormente devolver una respuesta o una actuación programada. La otra implica definir un webhook y dedicar un servicio propio a la atención de las solicitudes. En este caso se configura para que responda disponiendo llamadas a funciones que se ejecutarán en función del contenido de la petición que se reciba. De esta forma, empleando una solución asíncrona se logra una mayor eficiencia y se puede saber cómo llegan las peticiones al servidor y este responde, es decir, se pueden analizar las tramas enviadas y recibidas tanto sus cabeceras como su cuerpo. La diferencia se puede ver visualmente en la figura 2.7.

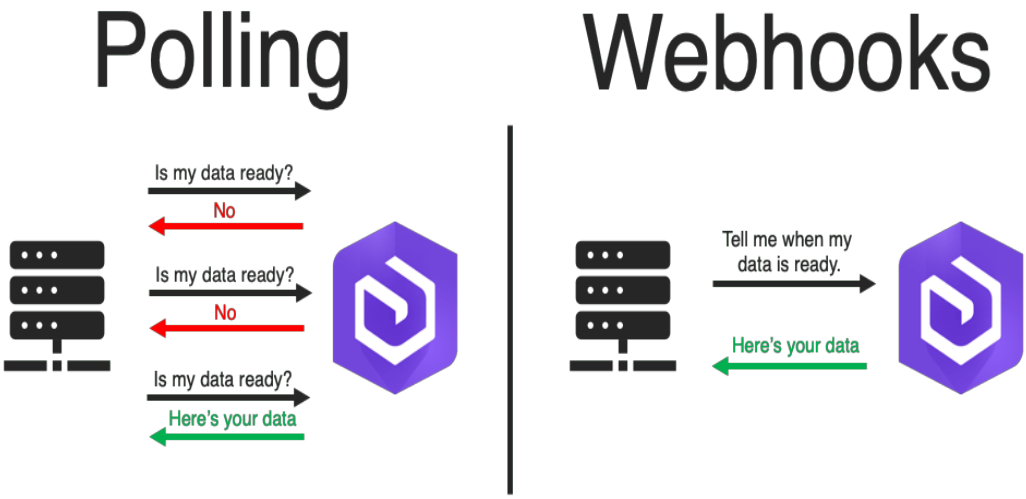


Figura 2.7: Polling vs Webhooks

La API de programación de bots de Telegram proporciona nuevamente dos tipos de funcionalidades. De una parte permite utilizar el bot como un Inicio de Sesión Único (SSO), a través del cual, tras iniciar sesión en la plataforma de Telegram, se obtengan garantías de identidad del usuario para posteriormente emplearlas para conectarse e iniciar sesión en páginas web o servicios externos mediante un token temporal recibido.

De otra, la API dispone los métodos, objetos[1], etc. para interactuar con el bot. Los métodos más característicos son *sendMessage, sendPhoto, sendDocument, sendAudio, sendAnimation, etc.*, se usan para enviar un mensaje a su destinatario correspondiente diferenciando su contenido (texto, audio, video, etc.) Independientemente del tipo de contenido los mensajes se pueden editar o borrar mediante *editMessage* y *deleteMessage*, respectivamente. Además, existen otros muchos métodos con finalidad de obtener información de usuario o intercambiar una localización, enviar encuestas, etc.

Ofrece también la creación de un monedero virtual. Telegram integra una plataforma de pagos, a través de Apple Pay o Android Pay, para realizar aportaciones monetarias por parte de los usuarios, creando un entorno de micromecenazgo (*crowdfunding*) en donde se pueda valorar económicamente el trabajo de los programadores del bot.

Puesto que un bot se comporta de forma similar a un usuario, se le pueden asignar permisos de administrador dentro de un grupo, supergrupo o canal. El bot podrá en tal caso degradar o promover usuarios, echarlos de grupos, realizar conteos de mensajes, fotos, etc.

Los objetos que se proporcionan como parámetros al recibir un mensaje en el bot son:

- *User*, el cual contiene información del propio usuario como nombre completo y nickname

(apodo) o si es un bot o no.

- *Chat*, en el que se incluye un identificador de chat, el tipo de chat (personal, grupo, supergrupo o canal), los permisos que se disponen, etc.
- *Message*, el cual contiene un identificador de chat de usuario, el tipo de mensaje, la fecha de creación, adjuntos del mensaje, etc.
- También pueden incluirse otros muchos para referirse a audio, vídeo, imágenes, localización, encuestas, etc.

Operativa de bots

La forma de interactuar con los bots, que no dejan de ser, aplicaciones de terceros integradas dentro de Telegram y aportando un servicio que no se podría de otra manera, todo ello controlado y mediado a través de la API anteriormente comentada, se explica de la siguiente manera:

Lo primero que se ha de hacer para crear el bot es configurarlo dentro de la plataforma de Telegram. A través de BotFather, un bot verificado propio de Telegram, se solicita la creación de un nuevo bot. Se especifica el nombre del bot, el nickname, la privacidad (permisos y accesibilidad a grupos), comandos posibles, foto de perfil, etc. Todos estos ajustes son opcionales excepto el nombre y el nickname. Una vez completada la solicitud, se devuelve el token único para acceder a ese bot, que habrá que vincular a la dirección del servidor. Todo esto se ampliará en detalle, junto con los parámetros introducidos para este proyecto en el capítulo *Creación y documentación del bot* 3.6.1.

Una vez creado un bot, se pueden emplear dos formas para interactuar con él. La forma más común de empezar la conversación entre el usuario y el bot es haciendo uso de comandos específicos. Por convención, cada comando se inicia con el carácter '/', seguido de sus parámetros particulares. Seguidamente la conversación se puede continuar aportando más datos después de haber introducido el comando mediante teclado o una botonera específica con 2 o más resultados predeterminados o la introducción de ciertos argumentos (números, palabras, claves, etc.)

Los mensajes intercambiados en el chat entre el usuario y el bot, por defecto se envían en el formato predeterminado (tamaño y tipo de letra) que tenga la interfaz en la que se ejecuta Telegram (al igual que ocurre en todos los chats). Sin embargo, también se puede remitir en formato enriquecido, de forma que se puede conocer, a través del análisis del contenido Lenguaje de Marcas de Hipertexto (HTML) con un método proporcionado por la API), el texto en negrita, cursiva, subrayado, resaltado de url, etc. Con esto, se consigue implementar una sentencia programada en HTML, con todas sus funcionalidades.

Lo último que se ha de reseñar de los bots es su funcionamiento global. Como se ha comentado en la API, se necesita que la programación esté ligada al token proporcionado por Telegram. A razón de esto, para que el bot sea funcional y responda a las peticiones, se necesita que esté en constante ejecución, ya sea a través de una máquina física o virtual, la cual, tiene que tener alojado el código y estar ejecutándose. De otro modo, el bot quedaría desconectado y no podría responder a ninguna petición ni mensaje o comando, ya que no tendría asociada la programación.

3 Bot de mensajería instantánea

A lo largo de este capítulo se realiza la creación del bot. Inicialmente, se expone el escenario básico del que surge el caso de uso del proyecto. Posteriormente, se detalla la arquitectura, tras lo cual se inicia la descripción de los desarrollos. Partiendo del entorno de desarrollo, se exponen a continuación los pasos seguidos en la creación del bot de gestión de Telegram aplicado al caso de uso del proyecto. Finalmente, se describen todas la funcionalidades y opciones implementadas, así como los algoritmos empleados para dar soporte a la generación del entorno seguro de gestión y distribución de claves.

3.1. Escenario

El proyecto se plantea con el objetivo de dar solución a la necesidad detectada de proporcionar una metodología sencilla y segura para dotar de acceso temporal a instalaciones privadas. Así enfocado desde el punto de vista de un residente de una vivienda, se busca poder dar permiso a diferentes personas para entrar a a su residencia, ya sean las zonas comunes o las zonas privadas. El método tradicional implicaría dejar las llaves a esa persona. Se busca evitar sistemas mecánicos y permanentes. Se presenta por tanto una solución en la que se asignen claves o códigos temporales que se introduzcan en un teclado o pinpad, y se verifiquen de forma automática garantizando el acceso seguro y autenticado a a la parcela deseada. El residente no se debe preocupar de que la persona pueda utilizar esa clave volátil en otro rango o circunstancias.

De esta manera, además de tener un sistema en principio más robusto, se puede llevar un control de la afluencia en determinadas zonas, al asignar claves diferentes para cada usuario y con un periodo de validez reducido.

Dentro de la plataforma, habrá dos tipos de usuarios dependiendo de sus permisos y limitaciones, clasificados en residentes y visitantes.

El residente tendrá la capacidad de crear nuevas claves temporales para usuarios externos y gestionarlas para en caso de que exista algún contratiempo pueda revocar su validez. Fuera del horario la clave será invalida. El visitante, durante cuyo registro se deberá contrastar y corroborar su identidad, podrá consultar la clave asignada a una entrada. Además en ambos casos se dispondrá de una ayuda que guie al usuario en los diferentes comandos y funcionalidades disponibles.

Todos los procesos se soportarán empleando una aplicación de mensajería instantánea. Será por tanto a través de mensajes de texto o multimedia que los usuarios y el sistema de gestión intercambiarán la información. En esta coyuntura, la solución confía en los mecanismos de confidencialidad e integridad propios de una aplicación de mensajería segura. Para el caso que nos ocupa, ésta es Telegram.

3.2. Plataforma de desarrollo

La programación del bot de Telegram se realiza sobre el entorno de desarrollo *Visual Studio Code (VSC)*, operando sobre el sistema operativo *Ubuntu*[2] *(Linux)*. Ambas plataformas son *open source*[3] por lo que no se requiere licencia.

Con este entorno, el proceso de desarrollo ha resultado más intuitivo, gracias en parte a las numerosas ayudas que implementan. Se ha empleado la plataforma GitLab para el seguimiento y gestión de las diferentes versiones del código.

3.3. Arquitectura

Una vez detallados los tipos de usuarios dentro del proyecto y sus funciones principales, se expone la arquitectura que va a tomar el proyecto. En este caso, la arquitectura a gran escala del proyecto queda realzada como se observa en la figura 3.1.

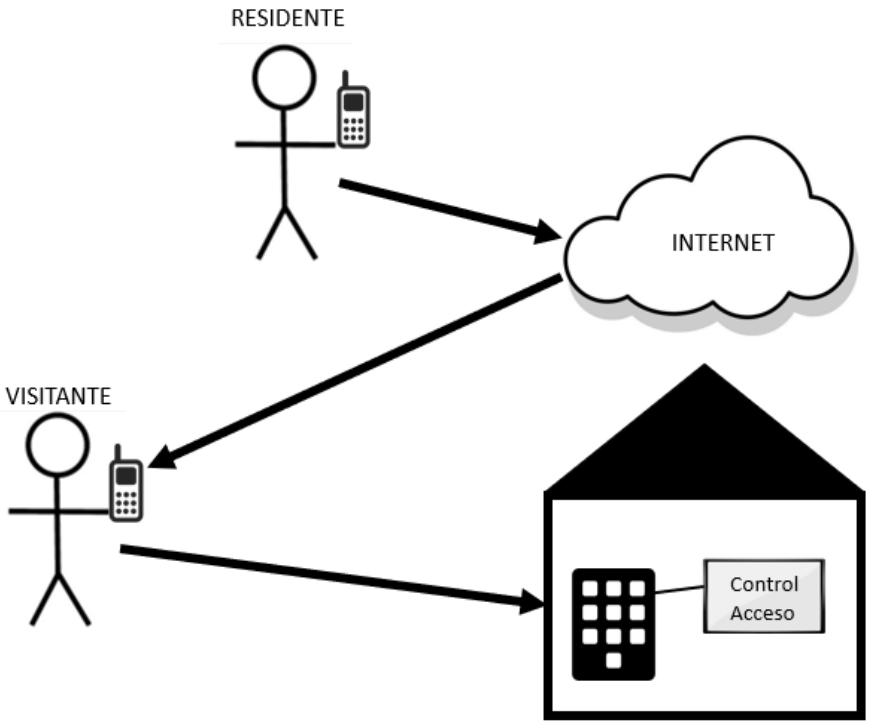


Figura 3.1: Arquitectura de alto nivel de la solución

En este caso, el residente, que es el primer interventor del proceso, se comunica con la aplicación de mensajería instantánea Telegram, mediante Internet. Se producirá el traspaso de datos correspondiente y cuando sea conveniente, estos datos les serán comunicados al visitante a través

de la misma aplicación. De esta manera, el visitante será el beneficiario de los datos y podrá continuar el proceso en el lugar donde estará el control de acceso para finalizarlo.

3.4. Máquina de Estados

Antes de iniciar el proceso de desarrollo, se llevó a cabo la especificación de la operativa deseada para el bot y el sistema de forma global.

La primera tarea que debe realizar el bot consiste en ratificar si la persona que quiere acceder al bot consta de permisos, para poder conceder o denegarle la capacidad de realizar las acciones anteriormente mencionadas. En la figura 3.2 se muestra la máquina de estados del bot para gestionar este proceso.

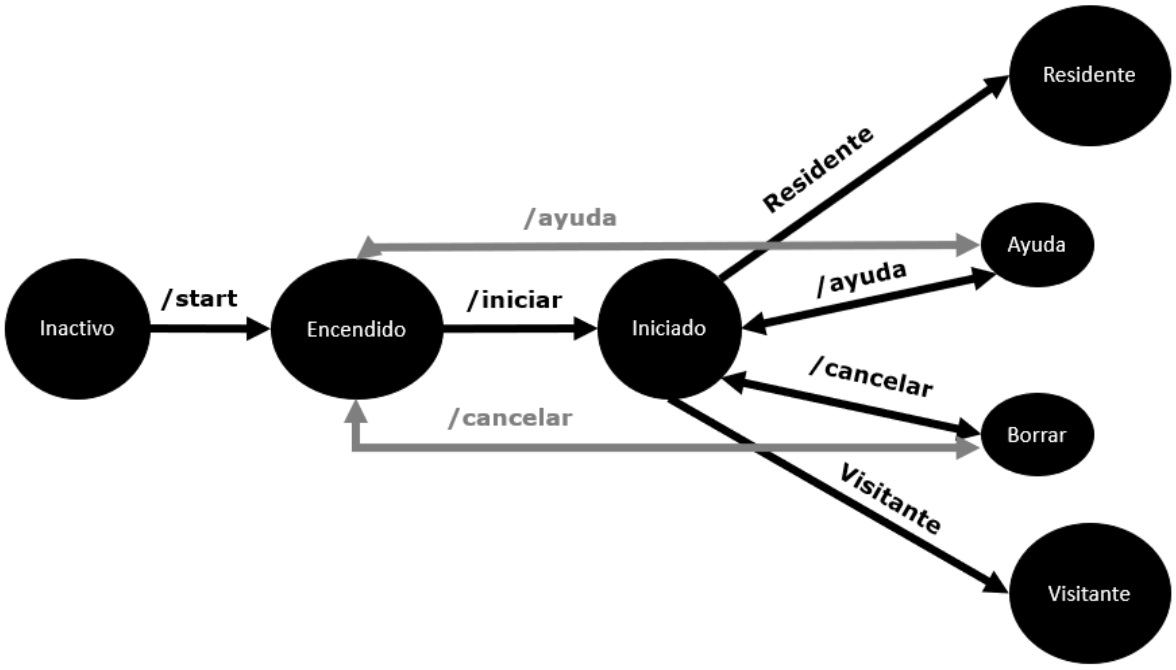


Figura 3.2: Máquina de estados del bot

Considerando que el bot no ha iniciado conversación alguna, pasará del estado *Inactivo* al de *Encendido* una vez se ejecute el comando */start*. En este estado, el bot será capaz de acceder a los estados de *Ayuda* y *Borrar*, los cuales ofrecerán una guía de apoyo y eliminan una clave de acceso emitida anteriormente, respectivamente. Una vez acabados estos procesos, vuelven al estado del que partieron.

El inicio de toda la ejecución de la lógica del bot se realiza al pasar al estado *Iniciado*, estado al que se accede mediante el comando */iniciar*. En él se presenta el proceso de elección del rol del

usuario que interactuará con el bot, debiendo escoger entre Residente o Visitante. Internamente, el bot controlará si el usuario con el que se comunica tiene los privilegios necesarios para acceder.

Si se opta por el papel de Residente y se ratifican los permisos para acceder, se inicia el proceso de creación de la clave temporal que se asignará al visitante. El proceso se describe en la figura 3.3.

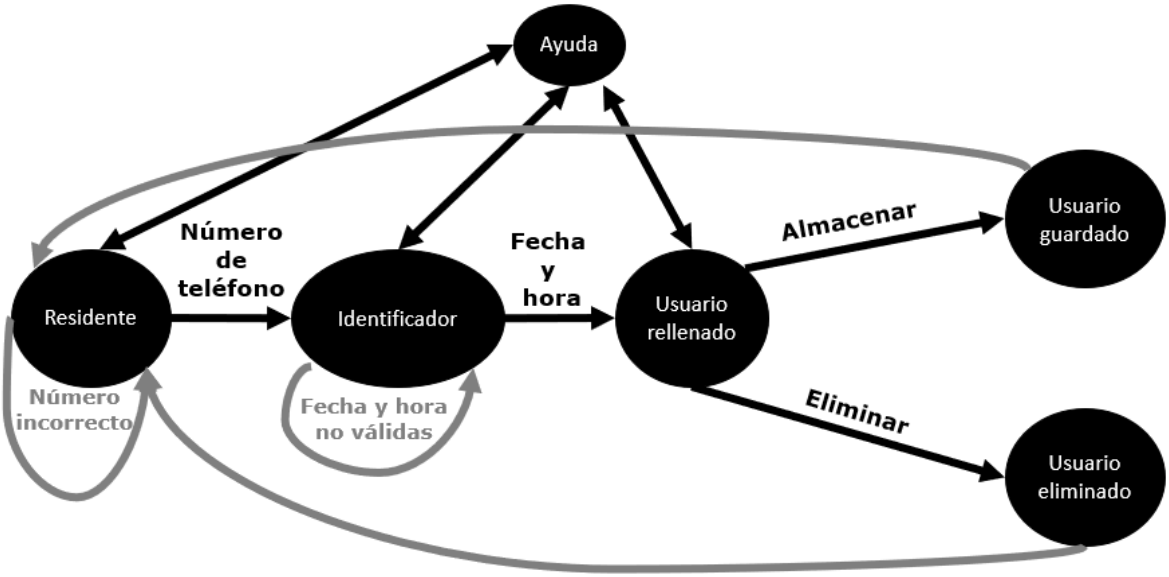


Figura 3.3: Máquina de estados del bot para el rol de *Residente*

Una vez en el estado *Residente*, se deberá introducir un número de teléfono válido, que servirá de identificador del visitante o entidad para la que se está generando la nueva clave de acceso. Para establecer las restricciones iniciales de validez de la clave se deben introducir una fecha y hora posterior a la actual. Si se introducen valores que no concuerdan con el tipo de dato esperado lo que se piden, se retorna al estado del que parte. Una vez que se hayan introducido correctamente estos datos, se tendrá la posibilidad de almacenar esos resultados o descartarlos. Finalizado este proceso, se volverá al estado inicial. En cualquier momento del proceso, se podrá consultar la ayuda por si existe alguna duda de cómo continuar con el proceso.

Para la opción de Visitante, la máquina de estados resultante se muestra en la figura 3.4.

Una vez en el estado *Visitante*, se introduce un número de teléfono válido y se pasa al siguiente estado *Usuario rellenado*. Ahí se verificará si esta información introducida se corresponde con algún usuario guardado. Si no es así, se retornará al estado inicial y si existe alguno, se devolverá la información necesaria para que el visitante pueda tener acceso a la instalación o recinto. Como en el caso anterior, durante todo el proceso se podrá consultar la ayuda por si hay alguna duda de cómo proceder.

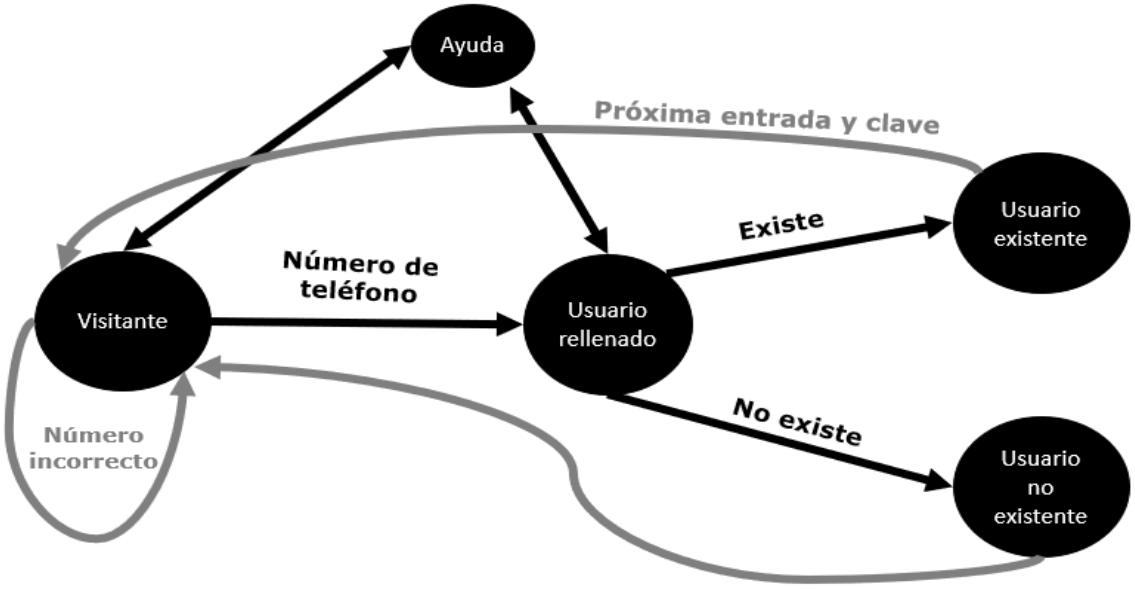


Figura 3.4: Máquina de estados del bot para el rol de *Visitante*

Todo su desarrollo en detalle, así como todas las posibilidades que hay para completar correctamente los ciclos, junto con las ventanas de funcionamiento para obtener el fin deseado, se remarcará en el apartado 3.6.

3.5. Evaluación práctica del API

Una vez definida la operativa básica del bot, llega la hora de comenzar su implementación. Como primer paso se realiza una toma contacto con la API de Telegram y el proceso de creación y operativa de un bot.

Para evaluar la funcionalidad del bot, se desarrolló un sencillo bot que sirviera de base sobre la que posteriormente añadir funcionalidades complejas. En este caso, se implementó un bot que solamente reaccione a 2 comandos e ignore el resto. Como se ha descrito anteriormente, este bot deberá ser registrado con un alias obteniéndose su correspondiente token.

El bot actúa de la siguiente manera, tal como se muestra en la figura 3.5:

- Cuando se introduce el comando '/abrir', el bot responde con el mensaje: "Abriendo la puerta"
- Cuando se inserta el comando '/cerrar', el bot responde con el mensaje: "Cerrando la puerta"

- Cuando se escribe por pantalla cualquier otra cosa, o se selecciona otro comando inexistente, el bot responde con el mensaje: “comando incorrecto”

Figura 3.5: Menajes comando-respuesta en un bot de prueba

Este bot implementa una secuencia de comandos-respuestas, donde un usuario manda un mensaje y el bot le responde. No obstante, ha servido para ver cómo emplear la API de Telegram adaptada al lenguaje Node JS. Tras obtener el token del bot, previo registro a través de BotFather, se ha desarrollado un conjunto de funciones que respondan ante los eventos generados por los comandos descritos. Este trabajo ha permitido conocer los valores que los parámetros de las funciones reciben y analizar cómo trabajar con ellos para lograr el objetivo del proyecto.

Se han utilizado varios parámetros de configuración del bot como son el servidor (en este caso viene incluida con la API), los ’workers[4]’ utilizados, promises[5], la ejecución de ’polling’, etc. Esto se observa en el código 3.1.

```
1 tlg.TelegramBot = require('node-telegram-bot-api'); //API bot
2 tlg.bot = new tlg.TelegramBot(tlg.token, {              //options bot
3    workers: 1,async: true,  //trabajadores y modo no sincrono
4 },);
5 tlg.bot.startPolling();  //comienzo de polling
6 tlg.bot.pollingTimeout=60;  //tiempo max de respuesta
```

Código 3.1: Parámetros de configuración del bot con polling

La ejecución del bot permite observar el nulo retardo entre el envío del comando y la recepción de la respuesta, demostrando además que para un usuario convencional la conversación mantiene los parámetros y metodología a la que ya está acostumbrado.

Este bot no muestra todos los aspectos que se han implementado posteriormente en el bot final, como la asignación de un webhook, el registro al bot mediante un token o clave, la utilización de variables dentro de los mensajes, tanto de entrada (aportados por el usuario), como de salida (generados por el bot), el uso de una botonera predefinida, etc. Sin embargo, ha servido como base de pruebas para muchos de esos desarrollos. La configuración del webhook y demás parámetros dentro del bot hace que la programación quede tal como se refleja en el código 3.2

```
tlg.TelegramBot = require('node-telegram-bot-api'); //API bot
tlg.bot = new tlg.TelegramBot(tlg.token, {      //options bot
   webHook: {         //creacion de webhook
      url: 'https://ngrok.io',  //servidor externo
      port: 5443,            //puerto
      host: 'localhost'      //host
   }, workers: 1, async: true, params: { //resto configuracion
      timeout: 60
   }
},);
```

Código 3.2: Parámetros de configuración del bot con webhook

3.6. Implementación

A continuación se van a exponer todos los detalles y el funcionamiento del bot. Primero se va a precisar la generación del bot, junto con su documentación. Seguidamente, se enumerarán los diversos comandos de acuerdo a la máquina de estados detallada anteriormente. Por último, se hará un repaso de todo ello, y así especificar la base de datos necesaria para almacenar toda la información resultante.

Además hay que añadir que para que el bot responda cuando se le escribe o rellena un comando tiene que estar activo. Para ello, el programa del bot debe estár ejecutandose constantemente, para lo que se dispone alojarlo en un servidor. La opción más habitual hoy en día es emplear un entorno virtualizado en la nube.

3.6.1. Creación y documentación del bot

Lo primero que se ha hecho es acudir a BotFather dentro de Telegram para asignar un token específico al bot, tal y como se había descrito anteriormente. En este caso, el bot se ha registrado bajo el alias *@controlpuerta_bot* (el fin del alias *_bot* es para diferenciar los alias de usuarios reales de los bots) y con el nombre de *Control de acceso SerCa*. Se le ha añadido una descripción para que los usuarios comprendan mejor la operativa del bot y, en caso de interés, se comuniquen con el. Así, se indica que el "bot es capaz de general claves temporales o visionarlas

para posteriormente ser introducidas en un pinpad y acceder a un lugar determinado.". Se incluye también un foto de perfil, resultando la presentación del bot que se muestra en la figura 3.6.

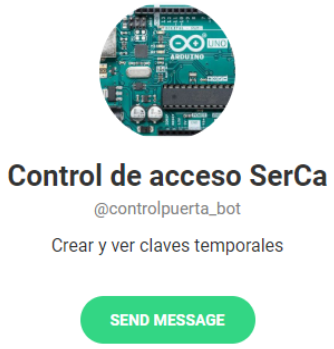


Figura 3.6: Registro de usuario en el bot

Entre los detalles que se indican a la hora de registrar el bot, se ha especificado que el bot no se pueda introducir en grupos o que no pueda ser mencionado en conversaciones, tanto grupos, supergrupos, canales o chats individuales. Por último, no se ha añadido a la información del bot ningún método para recibir algún tipo de monetización disponible, si bien se podría haber vinculado a un pago por uso de generación de códigos.

A continuación se enumeran los comandos correspondientes que se van a utilizar y que se describen en secciones posteriores. Estos comandos son */iniciar*, */cancelar* y */ayuda*. Estos se ven representados en la figura 3.7. Para introducirlos se puede escribir el comando entero o pinchar en la parte del desplegable del comando correspondiente.

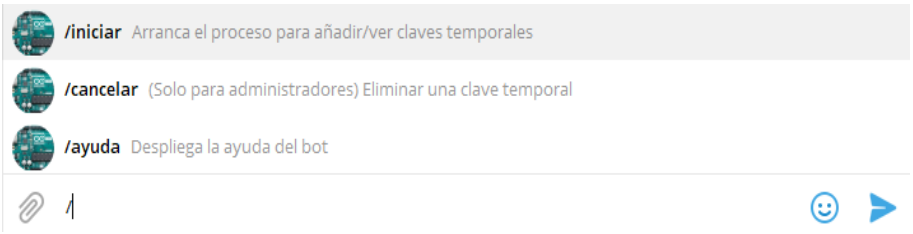


Figura 3.7: Comandos para interactuar con el bot

3.6.1.1. Funcionamiento del servidor

El bot no está directamente unido entre cliente y servidor de Telegram, sino que pasa por medio de un webhook, como se han explicado anteriormente. En este caso, el webhook se sitúa entre los clientes que deseen acceder al bot y el servidor de Telegram que gestiona los accesos al bot. Posteriormente, el servidor propio de Telegram recoge las peticiones a través de su API y mediante el token y el identificador remite las respuestas.

Mediante la implementación de este webhook, se pueden ver todas las peticiones que realiza cada usuario. Como se ha comentado, todos los datos se encapsulan en peticiones HTTPS de tipo POST (junto con el identificador del bot y el token) y se estructuran en formato Notación de Objeto de JavaScript (JSON). El intercambio de una trama se puede observar en la imagen 3.8.

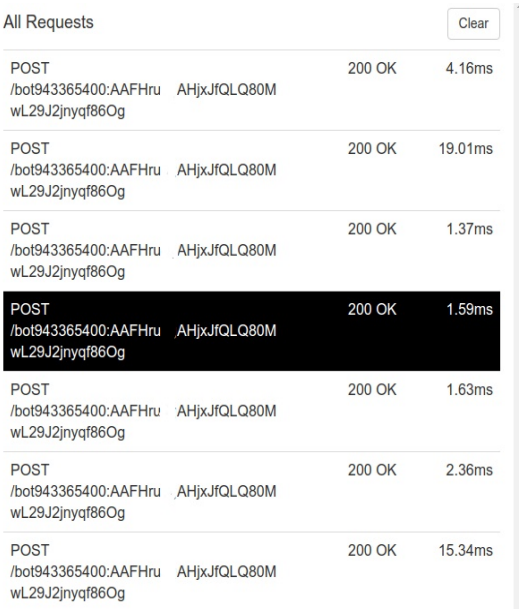


Figura 3.8: Peticiones 'POST' recibidas por el webhook

El cliente cuando escribe un mensaje o pulsa un botón dentro del bot, crea un mensaje JSON con esos datos junto con sus identificadores, que se envían al webhook. Además, esta petición debe ir acompañada de un identificador, que determine qué bot le está mandando la petición. Por último, se debe acompañar del token del bot asignado al usuario, que hará de verificador. Este validará si el identificador del bot concuerda con los parámetros asociados al token. Además, dentro de la petición JSON se envía el identificador de mensaje que se está escribiendo Un ejemplo de una petición JSON es la representada en el código 3.3.

```json
{
   "update_id": 397009634,
   "message": {
      "message_id": 8228,
      "from": {
         "id": 383081345,
         "is_bot": false,
         "first_name": "Sergio",
         "last_name": "Castillo",
         "username": "sergiokrak",
         "languaje_code": "es"
      },
      "chat": {
         "id": 303181345,
         "first_name": "Sergio",
         "last_name": "Castillo",
         "username": "sergiokrak",
         "type": "private"
      },
      "date": 1584852436,
      "text": "/start",
      "entities": [
         {
            "offset": 0,
            "length": 6,
            "type": "bot_command"
         }
      ]
   }
}
```

Código 3.3: Mensaje JSON intercambiado entre el cliente y el webhook

Una vez que el webhook reciba las peticiones HTTPS, las analiza y las reenvía al servidor de bots de Telegram aportado todas estas configuraciones del bot mencionadas. De esta manera, el servidor de bots de Telegram recibe toda la información necesaria y el bot correspondiente puede realizar la funcionalidad a desarrollar. Una vez que la petición llega correctamente, el servidor de bots de Telegram envía el código *200 OK* de respuesta, pasando por el webhook hasta finalmente acabar en el cliente. Este código significa que todo está correcto y se va a realizar la función indicada. Esta función a realizar no se envía como respuesta desde el servidor de bots de Telegram hacia el cliente, sino que una vez que el servidor disponga de la información necesaria, realizará las operaciones necesarias y remitirá al bot el mensaje o la acción correspondiente, por lo que el cliente lo percibirá directamente en el chat. Ese mensaje que emite el bot va dirigido hacia los servidores centrales de Telegram que gestionan todo el intercambio de información. Este proceso puede verse reflejado en la siguiente imagen 3.9.

3.6.2. Inicialización del bot

Para que un usuario tenga plena capacidad de actuación con el bot, siguiendo la máquina de estados planteada, se debe clasificar al mismo dentro del bot. El primer paso es acceder al

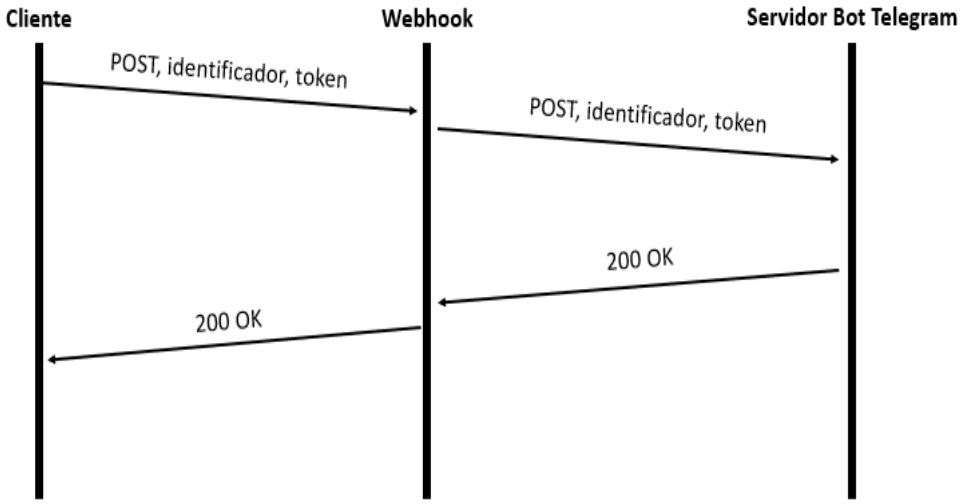


Figura 3.9: Intercambio de mensajes entre cliente y bot

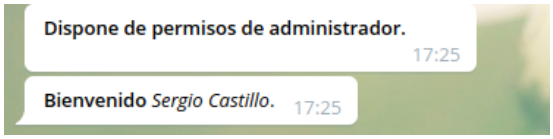
bot, para lo cual Telegram facilita la creación de URL que proporciona acceso directo a cualquier usuario, en este caso al bot. Así para nuestro caso particular el enlace proporcionado es de la forma *https://t.me/controlpuerta_bot*. El enlace que se genera para los propietarios y para los visitantes es diferente en cuanto a los parámetros que lleva asociados el enlace. Respecto a los visitantes, el enlace se mandará tal cual se ha expuesto anteriormente, pero en el caso de los residentes llevará añadido una clave maestra como parámetro para indicar que es residente. El formato será *https://t.me/controlpuerta_bot?XXXXXXXXX*, donde 'XXXXXXXXX' será la clave maestra numérica correspondiente.

También existe la posibilidad de utilizar el buscador de Telegram, donde introduciendo `'@controlpuerta_bot'` se puede identificar al bot e iniciarle. Sin embargo, de esta manera no se estaría introduciendo ninguna clave maestra de acceso, ya que se hace sobre la misma interfaz de Telegram, por lo que este método sólo es recomendado para los usuarios visitantes que deseen interactuar con el bot.

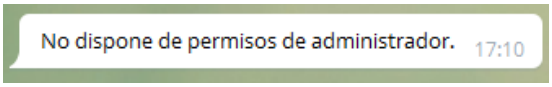
Una vez se accede a dicho enlace o se selecciona el chat del bot correspondiente, automáticamente se redirige a la interfaz de Telegram, concretamente al chat con el bot, donde se lanza el comando `/start` para activar la conversación entre el bot y el usuario.

3.6.3. Uso de los comandos

Una vez que se ha habilitado la conversación con el bot por parte de un usuario, el bot está totalmente operativo para dicho usuario y expone todas sus funcionalidades. Como primer paso, se ha implementado un saludo o mensaje de bienvenida acompañado de una aclaración de si tienen o no permisos para realizar las funciones de residente. En la figura 3.10 se muestran unas capturas de dichos mensajes.



(a) Bienvenida



(b) Permisos

Figura 3.10: Mensaje de bienvenida y aclaración de permisos disponibles

Pasado este proceso de saludo, el usuario ya estará en plena posibilidad de introducir cualquiera de los 3 comandos posibles, salvaguardando los privilegios disponibles. A continuación se explicará el funcionamiento de cada uno de los comandos, cómo debe de usarlos un usuario y a qué partes puede o no acceder según al tipo de usuario que pertenezca.

3.6.3.1. /iniciar

Este es el comando principal, el cual inicia el proceso de creación o visionado de la clave. Cuando lanza este comando se acciona un desplegable con una botonera, donde se debe seleccionar un tipo de usuario (figura 3.11).

Figura 3.11: Botonera de elección de tipo de usuario

Si dispone de los permisos necesarios podrá acceder a los 2. El tipo `Residente` dará la posibilidad de generar una nueva clave temporal asociada a un visitante, mientras que el `Visitante` guiará al usuario externo hasta obtener la clave que deberá introducir para su próxima visita, junto con información adherida.

Si el usuario con el que conversa el bot no tiene permisos, es decir, es un usuario visitante y selecciona el botón `Residente` se le indicará que no dispone de permisos y el bot se quedará a la espera un nuevo proceso en donde se seleccione un tipo de usuario permitido tal y como se aprecia en la imagen 3.10b.

A continuación se detalla el proceso implementado para cada uno de los tipos de usuario del bot.

Residente

Al seleccionar usuario `Residente`, lo primero que se va a comprobar es si realmente el usuario dispone de permiso. Una vez se confirme, se inicia el proceso de generación de las credenciales de acceso para un usuario visitante. Se deberá introducir el número de teléfono de la persona para la generará la clave temporal. Este número, servirá como identificador único del usuario, y deberá ser introducido posteriormente en el equipamiento de validación instalado en la localización securizada, tal como se describirá en la sección 4.4. Posteriormente, habrá que adjuntar la fecha y hora a la que va a acceder. Se ha establecido un intervalo de una hora para ingresar como valor por defecto para facilitar el proceso. Se considera que esta duración de clave presenta un compromiso entre riesgo y usabilidad. No obstante, si se observara demanda se podría establecer un periodo más corto o incluso fijar la duración a valores concretos. Por el momento, si se requiere un código temporal más extenso, se pueden generar más de un código para cada usuario, que el usuario percibirá debidamente (cada uno de 1 hora de duración).

En este ejemplo, se va a usar como identificador el número de teléfono: '694637845', la fecha que se le va a introducir es el día 22 de octubre del año 2020 a las 11 de la mañana. Este proceso se puede ver detallado sobre una interacción real en la figura 3.12.

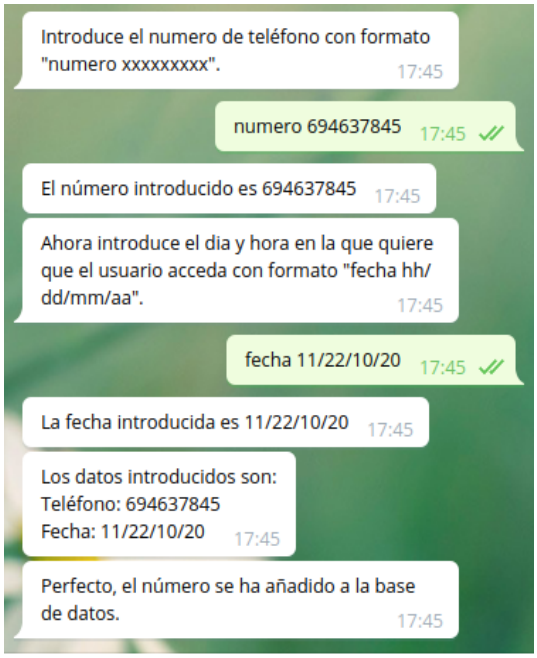


Figura 3.12: Ejecución correcta de un proceso de generación de una clave

El procedimiento para generar el código temporal se basa en el uso de Contraseña de Un Solo Uso Basada en HMAC (HOTP). Con todos estos datos introducidos, se procede a ejecutar un Hash MAC (HMAC) (SHA1) y posteriormente se aplica ese HOTP en donde se realiza el algoritmo que genera un token de 6 dígitos, los cuales formarán la clave temporal. La parte

técnica más detallada se explicará eb la sección 3.7. Esta clave generada, se almacenará en una base de datos y permanecerá ahí para que sea consultada por el visitante, la cual será detallada en la sección 3.6.4, a no ser de que la elimine el administrador mediante el uso del comando `/cancelar` como se detallará en la sección posterior o se retracte el usuario al final.

Durante el proceso de generación de la clave, una vez que se ha introducido toda la información necesaria, se le dará la posibilidad al usuario de confirmar la activación de la clave, que se almacenará en un base de datos con toda la información relacionada. En la figura 3.13 se muestra la captura de la pregunta de confirmación de entrada de los datos y en la figura 3.12 se puede observar el caso en que se solicita la activación de la clave.

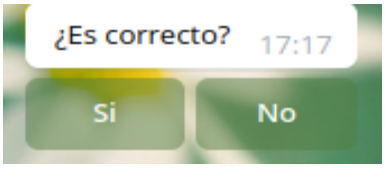
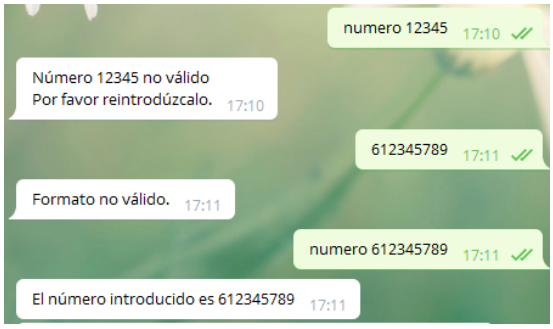


Figura 3.13: Mensaje de confirmación de datos introducidos

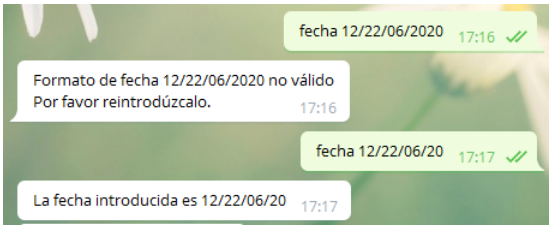
Cuando se está introduciendo tanto el número de teléfono o la fecha y se equivoca en un número, siempre se podrá reintroducirlo. Por ello el bot te devuelve "El número introducido es XXXXXXXXX", donde XXXXXXXXX sería el teléfono que se ha introducido. Volviendo a escribir de nuevo "numero YYYYYYYYY", donde YYYYYYYYY es el nuevo número corregido se sobrescribirán los datos. Esto sucede al igual que cuando se introduce la fecha.

Si se está introduciendo el número telefónico y se ponen de más o dejan de poner una o más cifras al introducir un número telefónico, el bot mandará un mensaje diciendo: "Número no válido" y le pedirá que lo reintroduzca. Si por otro lado, se está introduciendo el número y no se expecifica correctamente del tipo "numero XXXXXXXXX" sino solo "XXXXXXXXX" se notificará con el mensaje "Formato no válido" y el bot esperará a que se corrija el fallo y se introduzca correctamente. Este mensaje también se notificará si está en una fase en la que se requiere que se introduzca un dato y se introduce uno diferente, como por ejemplo se pide que se introduzca el teléfono y se introduce la fecha y hora. Si el usuario se percata del error y envía de nuevo el mensaje corregido, el bot lo acepta, almacena ese dato y pasa a la siguiente fase.

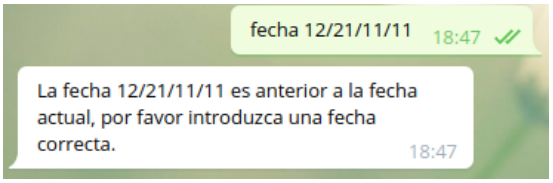
Al introducir la fecha se requiere que los parámetros se introduzcan entre barras '/' y los datos temporales todos de 2 cifras. Si esto no se cumple el bot lo detectará y se remitirá el mensaje de "Formato de fecha no válido" y se requerirá que se vuelva a introducir. Además, como es obvio, no se van a generar claves para acceder en tiempo pasado, es decir, que la fecha sea anterior a la actual. Si esto es así se notificará para que se reintroduzca ya que de otro modo no tendría validez alguna.



(a) Error número no válido y formato incorrecto



(b) Formato de fecha incorrecto



(c) Error fecha introducida anterior a la actual

Figura 3.14: Errores surgidos con la interacción del bot

Visitante

Si se selecciona el usuario *Visitante*, el cual no requiere de permisos de administrador, lo que se le va a pedir es que introduzca el número de teléfono (identificador) del Visitante. De esta manera se verificará si la persona es apta para acceder. Una vez introducido se le mostrará la próxima entrada, aunque puede tener también información vinculada a permisos de acceso posteriores.

En un mensaje, como se observa en la figura 3.15, se le mostrará la clave temporal con la que podrá acceder y seguidamente en un segundo mensaje se indicará la fecha y hora de esa próxima entrada en formato Esta información irá acompañada de la duración de la clave temporal (1 hora como ya se ha justificado). Esta imagen se corresponde con la continuación del ejemplo anterior (mismos datos de entrada). En este caso, el visitante desea consultar la entrada anteriormente guardada por el residente. Introduce el identificador y se le muestra la clave: '220079' y el horario de su entrada como se había indicado: el día 22 de octubre del año 2020 a las 11 de la mañana.

Si se introduce un número de teléfono válido, es decir, un identificador que cuenta con sus características, 9 dígitos numéricos, pero no ha sido registrado a través de un residente, para aportar ninguna clave temporal se le notificará al usuario según se muestra en la figura 3.16. Tras ello, la ejecución se detendrá y el bot esperará que se reanude otra vez el proceso con la ejecución del comando `/iniciar`, donde el visitante deberá seleccionar otra vez su casilla e introducir un número válido registrado.

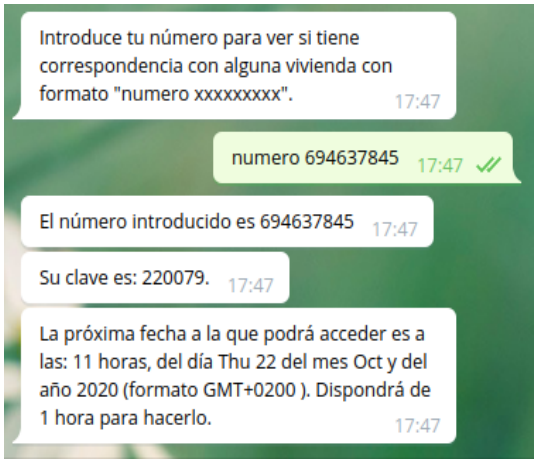


Figura 3.15: Ejecución correcta de un proceso de visionado de una clave

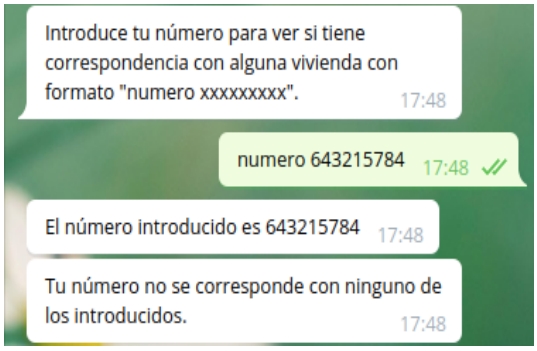


Figura 3.16: Ejecución incorrecta de un proceso de visionado de una clave

3.6.3.2. /cancelar

Gracias a este comando y si se disponen de los permisos apropiados, se podrá revocar una clave de acceso. Este comando se ha de usar cuando se ha añadido una clave a la base de datos y ya no se quiera usar en la situación prescrita. No obstante, siempre se tiene la posibilidad de cancelar durante el proceso de creación.

Dada la relevancia de la operación, el comando solicita confirmación explícita para ejecutarse e incluye una explicación acorde indicando que los datos no podrán ser recuperados, etc.

Si se confirma el borrado se le mandará un mensaje confirmando la eliminación de toda la información referente (figura 3.18a). Si por el contrario, se retracta y no se decide eliminar se especifica otro mensaje donde se expone la no eliminación y conservación de los datos (figura 3.18b). A esto, se le añade otro mensaje en el que se explica que como ha eliminado una clave, si luego quiere volver a generar otra, siempre puede usar el comando */iniciar* y repetir el proceso para crear otra clave (puede contener incluso el mismo usuario y fecha y hora en caso de que se haya borrado por equivocación).

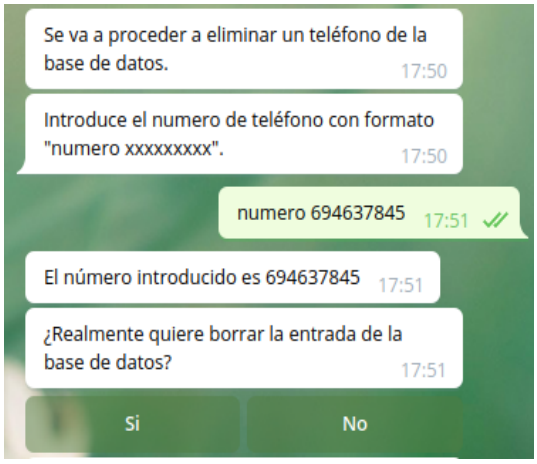
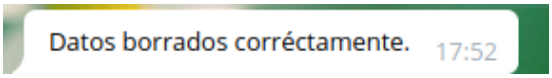
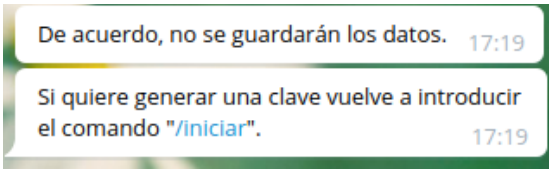


Figura 3.17: Ejecución correcta de un proceso de supresión de una clave de la base de datos



(a) Datos eliminados de la base de datos



(b) Datos no eliminados de la base de datos

Figura 3.18: Interacción del bot con la base de datos

3.6.3.3. /ayuda

Este comando sirve como apoyo para enseñar a cualquier tipo de usuario que quiera usar el bot los comandos posibles y su utilización. Puede ser usado en cualquier momento.

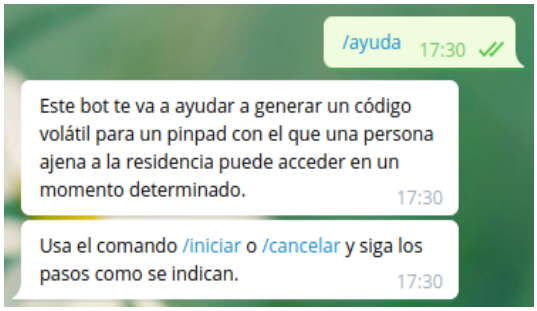


Figura 3.19: Respuesta al comando /ayuda

Como se aprecia en la figura 3.19, muestra una pequeña explicación descriptiva de la funcionalidad del bot y señala los comandos disponibles.

3.6.4. Base de datos

Todos los datos que se van introduciendo mediante el bot de Telegram son almacenados para sus posteriores consultas en una base de datos.

La base de datos se implementa usando MySQL, uno de los sistemas más extendidos de gestión de bases de datos relacionales. Haciendo uso de una librería específica para NodeJS, se ha adaptado la programación para poder acceder a la base de datos y realizar las interacciones necesarias entre los dos lenguajes. Gracias a esto, se pueden registrar todas las nuevas entradas directamente, al igual que borrarlas, o consultarlas realizando un barrido de datos para obtener la información de la tabla.

Para la correcta ejecución y almacenado de los datos necesarios se ha generado una tabla con las siguientes columnas, y las filas corresponden a cada entrada introducida. Los apartados correspondientes se pueden observar en la tabla 3.1.

Dato	**Formato**	**Cantidad de dígitos**	**Obligatorio**
tlf_id	Entero autoincrementado	∞	Sí
tlf	Número telefónico	9	Sí
clave	Integer	6	Sí
hora	Fecha y hora (Unix Timestap)	10	Sí
nombre	Char	0-255	No

Tabla 3.1: Descripción campos SQL

Los campos `tlf_id, tlf, clave` y *hora* son obligatorios y deben estar disponibles para una correcta operativa del bot. Por tanto, cada entrada en la base de datos estará formada por estos campos. El campo `tlf`, no es más que el número de teléfono que se ha utilizado para registrar al usuario. Este se almacenará en formato numérico de 9 dígitos. El tercero, `clave`, es la clave temporal resultante de todo el proceso de generación de la misma. Esta tendrá una longitud de 6 dígitos en formato numérico. El cuarto, `hora` es la fecha y hora de entrada, a partir de la cual se le dará acceso al usuario. Este se almacenará en la tabla con formato Unix[8]. El último, `nombre`, es un identificador alfanumérico de quien ha registrado el número. La persona que genera las claves temporales para el resto de usuarios tiene generalmente asignado un alias (a veces no se tiene configurado). Si es así se introduce ese alias en esta columna para referenciarlo. Si resulta que la persona que genera esa clave temporal no tiene asignado ese alias, simplemente se queda ese campo en blanco.

Con todos estos datos necesarios comentados, se genera esta tabla, la cual estará almacenada junto al servidor del bot de Telegram, ya que los dos actúan conjuntamente y son intrínsecamente necesarios.

```
1    CREATE DATABASE telefonos;
2
3    USE telefonos;
4
5    CREATE TABLE telefono(
6       tlf_id smallint (5) unsigned not null auto_increment primary key,
7       tlf int (9) not null,
8       clave varchar (20) not null,
9       hora DateTime not null,
10      nombre varchar (20)
11   );
```

Código 3.4: Creación de la base de datos mediante sql

La programación necesaria para tener disponible la conexión del bot con la base de datos está reflejado en el código 3.4, donde se expone la creación (en lenguaje SQL) y en el código 3.5, donde se detalla la conexión.

```
1  const mysql = require('mysql'); //libreria
2  var connection = mysql.createConnection({     //opciones de la BBDD
3     host     : 'localhost',
4     user     : 'root',
5     //password : '',
6     database : 'telefonos' //nombre de la base de datos
7  });
8  connection.connect(); //conectar con la base de datos
```

Código 3.5: Parámetros de configuración de la base de datos en NodeJS

Una vez creada y conectada la base de datos, la forma para interactuar con ella es mediante comandos incluidos en la función que se puede observar en el siguiente código 3.6.

```
1  metedatos = function (base, persona) {
2
3     base.connection.query('INSERT INTO telefono(tlf,clave,nombre,hora) VALUES('+
          persona.num+','+persona.clave+',"'+persona.nombre+'",FROM_UNIXTIME('+
          persona.fecha+'))');
4
5     base.connection.end(); //fin de connexion
6  }
```

Código 3.6: Introducción de la información en la base de datos

3.7. HMAC, SHA1 y HOTP

Estos 2 algoritmos son los usados en el proyecto para realizar parte de la encriptación y generación de las claves temporales. Los dos están debidamente documentadas bajo las Petición de Comentarios (RFC) correspondiente. La RFC2104 [39], junto con la RFC4634 [40] y la RFC3174 [41], son las que conjuntamente determinan cómo se ha de utilizar el HMAC SHA1.

El SHA es una función de Hash[9] que se utiliza para 'resumir' los datos introducidos. En concreto el SHA1 genera una salida de 20 Bytes (40 caracteres hexadecimales). Gracias a esto se ha podido implementar el HMAC, que se rije bajo la siguiente fórmula matemática: $HMAC(K,m) = H[(K' \oplus opad) \mid\mid H((K' \oplus ipad) \mid\mid m)]$ y $K' = H(K)$ *"si K es más grande que la longitud de bloque", o K, "si no lo es"*. Donde K es la clave secreta, m es el mensaje, H es la función de hash (SHA1), *opad* es outer padding e *ipad* es inner padding. *Outer padding*, consiste en bytes con valor 0x5C expuestos repetidamente, e *inner padding*, consiste en una cadena de bytes con valor 0x36 concatenados. El operador '$\oplus$' significa Disyunción Exclusiva (XOR) y '||' representa concatenación. Con esto, se consigue generar códigos totalmente diferentes, dependiendo de los parámetros introducidos, los cuales también han de ser diferentes.

Para la implementación en cada tecnología se ha utilizado la librería correspondiente que mejor se adaptaba. Para su creación en Node JS, se ha utilizado un archivo subido a Github [43], como base sobre el cual, se ha modificado explícitamente para adaptarlo a las bases del proyecto, sólo con las partes intrínsecamente necesarias y eliminando redundancias y código no necesario para este fin. Por lo tanto no es una librería como tal, sino que es un archivo modificado que se ha introducido dentro del código del bot propiamente dicho.

Una vez hecho esto, se tienen los 40 caracteres hexadecimales, con los que se va a formar la clave temporal, en ambos sistemas. Para ello se ha utilizado un algoritmo denominado como HOTP, adaptado al proceso anterior (HMAC SHA1). Como su nombre indica, "Contraseña de Un Solo Uso Basada en HMAC", crea contraseñas únicas a partir de esos 20 bytes. Su proceso de reducción de ese hash en 6 caracteres numéricos decimales está realizado según la RFC4226 [45].

Para la implementación en Node JS se ha usado la librería "hotp" [46], la cual, se ha insertado en el apartado de "node_modules" donde están todas las librerías utilizadas. Configurando las opciones de configuración, se obtiene finalmente ese token de 6 caracteres numéricos, que va ser la clave temporal generada.

```
1 var hotp = require( 'hotp' );        //libreria hotp
2
3 func.redhash = function(hash) { //funcion realizar hotp
4    var hashred = hotp( hash, counter=4, { digits: 6 }) //6 digitos clave
5    return hashred; //devuelve clave
6 }
```

Código 3.7: Librería HOTP

Con este proceso se pasa de la información introducida por el usuario, a una clave de 6 dígitos numéricos decimales, idénticos en ambas interfaces para asegurar la veracidad de la clave temporal.

4 Equipamiento de validación

Tras describir el desarrollo y operativa del bot de Telegram de gestión, en este capítulo se presenta el equipamiento de validación instalado en la localización a securizar y que provisiona el acceso a la misma. El objetivo es desplegar un sistema que, en la medida de lo posible, actúe en modo no conectado para evitar costes de mantenimiento añadido motivados por el mantenimiento de una tarjeta SIM, etc. La interacción del dispositivo será únicamente con los visitantes, ya que será el mecanismo de acceso directo limítrofe que actuará como barrera, para conceder o denegar el acceso al servicio.

Tras una breve introducción en la que se describe el entorno hardware y software empleado, se detalla la máquina de estados y el planteamiento que se ha seguido para su implementación. Seguidamente se presenta el procedimiento de implementación, señalando como los diferentes módulos hardware se han integrado para dar forma a la plataforma objetivo. Se incluye también el desarrollo de los algoritmos software correspondientes. Finalmente, se concluirá con un estudio económico de esta parte en la sección 4.6.

4.1. Plataforma de desarrollo

Para implementar el dispositivo de validación se va a utilizar la placa Arduino Uno, así como varios módulo externos, tanto analógicos como digitales. Para la interacción directa con el usuario se empleará un teclado matricial y una Pantalla de Cristal Líquido (LCD). Será a través de estos elementos que el usuario introducirá su identificador único y la clave temporal asociada.

Puesto que la placa Arduino dispone únicamente de un reloj interno que se activa con alimentación, y para la validación de la clave temporal se requiere una referencia temporal absoluta, se incluye también un módulo RTC.

Para la controlar estos elementos hardware se han empleado librerías propias de cada módulo disponibles gracias a la comunidad de desarrolladores de Arduino.

La plataforma Arduino se programan empleando un lenguaje de alto nivel para microntroladores que en esencia recuerda a C++ [22]. El entorno de programación empleado ha sido el mismo que el utilizado para el desarrollo del bot. El entorno VSC dispone de una serie de plugins que permiten incorporar funcionalidades adicionales. En este caso se ha empleado el plugin[6] *PlatformIO*, que dota a VSC de plena compatibilidad con la Arduino Board [32].

4.2. Máquina de estados

En la figura 4.1 se muestra toda la máquina de estados en un único diagrama, donde se especifica toda la interacción que se ha de hacer con la plataforma y el resultado obtenido.

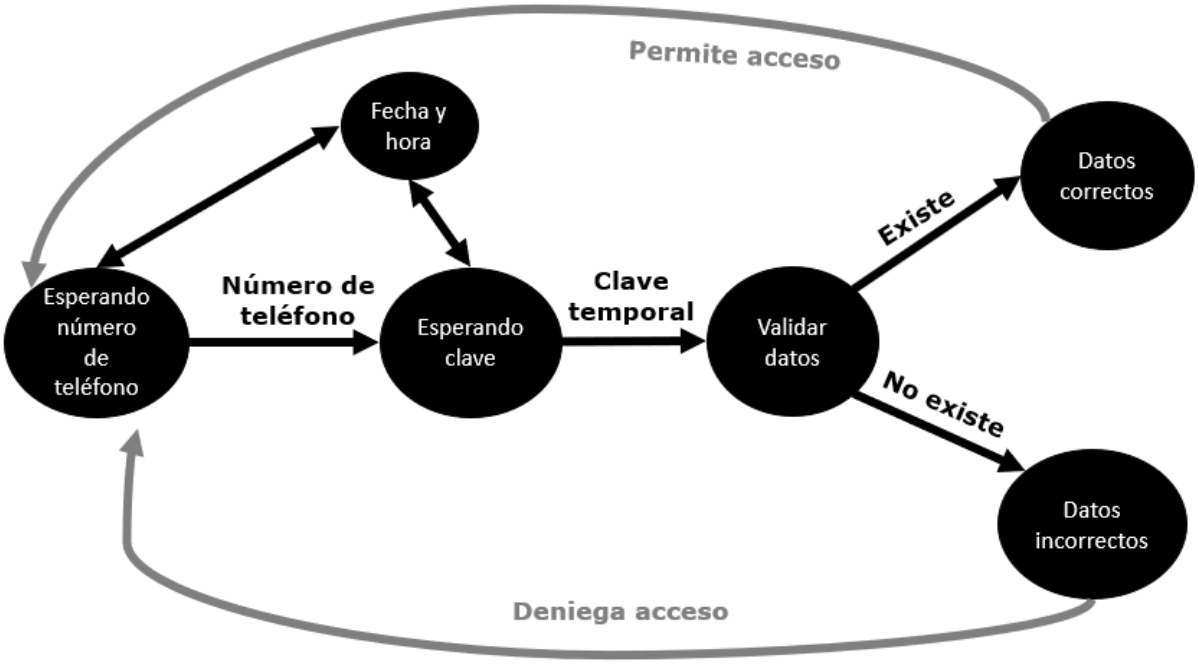


Figura 4.1: Arquitectura del sistema hardware Arduino

El sistema se encuentra por defecto en un estado de espera a la introducción del identificador de usuario, que en el caso de este proyecto es el número de teléfono. Una vez que se le introduzca este, se requiere se introduzca la clave temporal asociada al número de teléfono. Durante la estancia en estos dos estados (*Esperando número de teléfono* y *Esperando clave*), el programa automáticamente va a pasar al estado *Fecha y hora* cada segundo. Durante este momento, el programa actualiza su fecha y hora. Una vez se ha modificado, retorna al estado del que partió. Finalmente, cuando ya se han introducido estos datos, se validan y se corroboran. Si son correctos, se posiciona en el estado *Datos correctos*, donde se le concede el acceso y retorna al estado inicial. Si son incorrectos, se deniega el acceso y retorna al primer estado para que se vuelva a introducir los datos.

4.3. Algoritmo de validación

Este sistema, se puede dividir en dos partes, la propia implementación física, y la parte programable que hace de enlace y conexión entre la placa base y los diferentes módulos.

Como ya se ha indicado, la plataforma base para todo el desarrollo es una Arduino Uno. Este

sistema opera en modo mono-hilo donde de forma recurrente se llama a la misma función. Así pues toda la lógica de gestión de los módulos externos, como la propia de validación de los datos adquiridos se incluirá en dicha función, si bien existirán ciertas banderas que habilitarán unos códigos u otros para cumplir con la máquina de estados descrita en la figura 4.1.

El algoritmo de validación debe coincidir con el proceso que se ha implementado en el bot de Telegram. Por ello, todas las RFC correspondientes son las mismas que las especificadas en la sección 3.7. En líneas generales, los procesos de la generación de claves de Telegram explicados en el capítulo anterior y el proceso de validación de claves, llevado a cabo por Arduino son independientes. No existe ninguna red que conecte ambos procesos. Por ello, deben estar perfectamente coordinados los siguientes algoritmos junto con la fecha y hora del sistema.

Para facilitar la implementación se ha inyectado la librería *Cryptosuite2* [44]. Esta librería provee de varias funciones para realizar los algoritmos mencionados. La función principal incorporada viene adaptada para utilizar un SHA256 y no un SHA1, con lo que se han tenido que hacer ciertas modificaciones hasta obtener el algoritmo correcto. La implementación se puede observar en el código 4.1.

```
1  #include "sha1.h" //se coge solo el archivo sha1.h, no toda la libreria para
       que no haya colisiones
2  sha1_hasher_t sha = sha1_hasher_new(); //creacion sha1
3  sha1_hasher_init(sha); //inicializacon
4  sha1_hasher_init_hmac(sha,clave,10); //inyeccion de la clave
5  sha1_hasher_write(sha,msg,9); //inyeccion del mensaje
6  *result = sha1_hasher_gethmac(sha); //obtener resultado
```

Código 4.1: Establecimiento y ejecución de la librería Cryptosuite2

Antes de realizar la integración del hardware de introducción de los datos de usuario se ha afrontado el desarrollo de toda la lógica de validación de la clave temporal. Para ello se implementará la funcionalidad y se le introducirán los datos de usuario de forma estática.

Para evitar colisiones de código con otras librerías, solo se ha llamado al archivo *sha1.h* desde el programa principal. Este archivo lleva consigo la implementación del SHA1 y del correspondiente HMAC. Por ello, como se observa en el código anterior, se crea la estructura *sha*, donde se le va a introducir el identificador y la fecha y hora correspondientes. Posteriormente se realizan los algoritmos y se guarda el resultado en la dirección del puntero *result*.

Complementando la funcionalidad con la de librería *SimpleHOTP* [47] se logra obtener el algoritmo de validación operando de forma similar al que ya se había implementado para el bot. Esta última parte del algoritmo se puede ver implementada en el código 4.2.

```
#include <SimpleHOTP.h> //llamada a libreria
Key key(hmac, sizeof(hmac)-1); //formateo del hmac para adaptarlo al hotp
SimpleHOTP gen(key, 4); //general el valor HOTP
hotp = gen.generateHOTP();
return hotp; //devuelve el valor final
```

Código 4.2: Establecimiento y ejecución de la librería SimpleHOTP

Como se observa en el código anterior, la ejecución de esta librería completa el algoritmo desarrollado, al que se le introduce el HMAC obtenido anteriormente y se retorna el HOTP deseado.

4.3.1. Integración tiempo de reloj

Para poder ejecutar correctamente el algoritmo HOTP se require que tanto el bot como el equipo de validación tengan una misma referencia temporal. Puesto que la placa Arduino dispone únicamente de un reloj relativo a su puesta en marcha tras su conexión a la alimentación, se hace necesario incluir un módulo adicional que ofrezca esa funcionalidad.

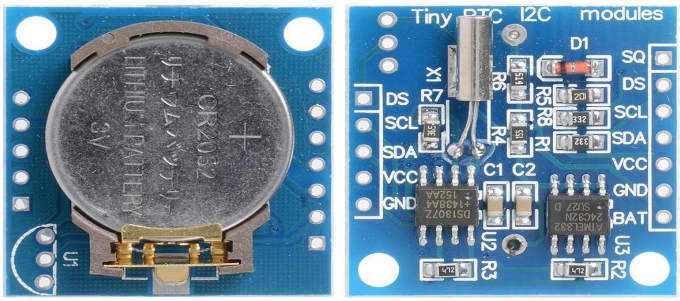


Figura 4.2: Módulo RTC DS1307

El módulo RTC DS1307 [49], mostrado en la figura 4.2, proporciona la operativa de un reloj absoluto. Funciona de manera independiente gracias a su pila de botón. Unido a un módulo I2C facilita su conexión a la placa Arduino. El conexionado requiere 4 pines, 2 para alimentación y otros 2 para datos analógicos. Los pines de alimentación no son intrínsecamente necesarios ya que se puede alimentar el módulo siempre de la pila de botón, sin embargo, para que no sufra un desgaste continuado, la corriente la suple la placa principal. En nuestro caso, la figura 4.3 muestra el resultado de la integración del módulo.

El módulo RTC necesita una corriente eléctrica continua, ya sea aportada por los pines de corriente de la placa base de Arduino o en su defecto por la pila de botón. Una vez establecida la hora, mediante el formato de tiempo Unix, actúa de contador a partir del momento que se desea. Por lo tanto, es capaz de mostrar la hora actual y permite interactuar sobre ella.

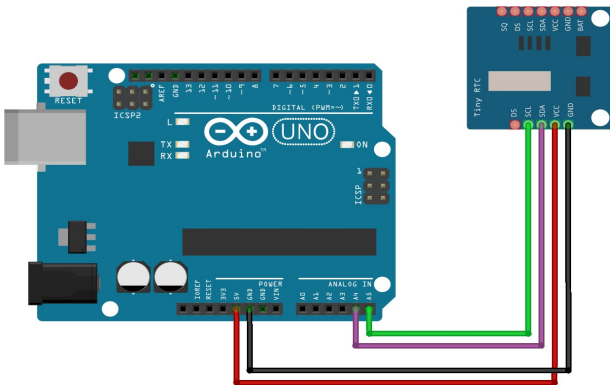


Figura 4.3: Módulo I2C para RTC

Para el uso eficiente del módulo se han utilizado 2 librerías para manejar de forma más sencilla la variable 'tiempo' y poder modificarla como mejor se corresponde tanto para lectura como para escritura. Por un lado, la librería DS1307 [49] que es la utilizada en el traspaso de variables desde el módulo a la placa. La otra librería, llamada Timelib [37] es la encargada de realizar las transformaciones necesarias para adaptar las variables a los formatos utilizados.

```
1   long int sacaunix(long int horamod)
2   {
3      minutos = minute()*60;
4      segundos = second();
5      horasobrante=minutos+segundos;
6      if (timeStatus() == timeSet) //si hay cambios
7      {
8         tac = now(); //tiempo unix long
9         horamod = tac - horasobrante; //tiempo formateado
10     }
11     return horamod;
12  }
```

Código 4.3: Obtención de la fecha y hora en formato Unix

Gracias a estas librerías, se muestran los siguientes códigos, el código 4.3, donde se formatea la fecha y hora hasta conseguir el formato Unix deseado y el código 4.4, donde se expone la fecha y hora de manera representable y legible.

```
1   void digitalClockDisplay(){
2      lcd.print(day());
3      lcd.print("/");
4      lcd.print(month());
5      lcd.print("/");
6      lcd.print(year()-2000);
7      lcd.print(" ");
8      lcd.print(hour());
9      lcd.print(":");
10     lcd.print(minute());
11     lcd.print(":");
12     lcd.print(second());
13  }
```

Código 4.4: Función de muestra del tiempo de forma legible

4.4. Interacción con usuario

El dispositivo hardware desarrollado hasta el momento requiere de otros módulos externos para habilitar la interacción con el usuario. Estos módulos serán el pinpad, un teclado físico a través de que se introduce información necesaria (clave e identificador), y la pantalla LCD, que será el elemento mediante el cual se reflejará la información de forma visual. Tanto los datos introducidos por el usuario como los procesados se mostrarán en ella a forma de verificación. De esta forma el usuario podrá ver gráficamente si puede o no acceder, o si ha introducido los valores que realmente deseaba.

Como entrada de datos manual se ha empleado un teclado matricial de membrana de 4 filas por 4 columnas como se refleja en la figura 4.4. Se trata de 16 botones (pulsadores), los cuales, al ser accionados, cierran esa parte del circuito, conectando la fila con la columna correspondiente. Cada fila y columna lleva asociado un cable, es decir, se dispone de 4 cables para las filas y 4 para las columnas. Estos se deben conectar a pines digitales (Modulación por Ancho de Pulsos (PWM)), en este caso se han elegido del 2 al 9, ambos inclusive. Se necesitan pines digitales debido a su lógica de funcionamiento. Normalmente, todas las conexiones están abiertas, pero al presionar un botón (pulsador), se conecta la fila y columna asociada, cerrando el circuito y mandando la señal correspondiente. Con este sencillo método, se mandan los pulsos necesarios, y así se pueden identificar los botones pulsados.

Para poder gestionar la información introducida, se ha usado la librería *Keypad* [35], la cual viene provista con varias funciones como el pulsado de una tecla. De esta manera, no hay que atender específicamente a cuando llega una señal digital e identificarla con el carácter correspondiente. Se ha realizado un mapa con los caracteres que se quieren obtener al pulsar la tecla, el cual se representa en el código 4.5.

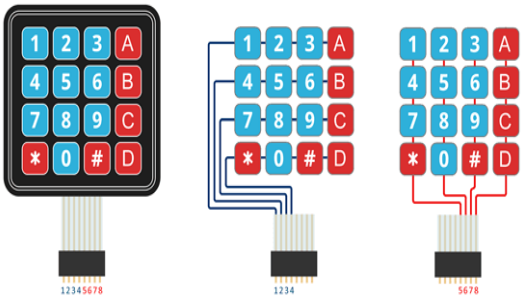


Figura 4.4: Teclado matricial de membrana 3x4

```
1   teclas[filas][columnas] = //definicin del mapa de caracteres
2   {
3      {'1', '2', '3', 'n'},
4      {'4', '5', '6', 'd'},
5      {'7', '8', '9', 'n'},
6      {'n', '0', 'n', 'd'}
7   };
8
9   Keypad teclado = Keypad (makeKeymap(teclas), pinsFilas, pinsColumnas, filas,
        columnas); //crear estructura Keypad
10  tecla=teclado.getKey(); //caracter obtenido
11  numeroteclado=tecla-48; //pasar caracter a numero
12  Serial.print(numeroteclado); //imprimir numero
```

Código 4.5: Definición del mapa de caracteres del pinpad y muestra de un carácter

Como elemento de visualización se ha seleccionado una pantalla LCD 1602 de fondo verde retroiluminado con letras negras con un tamaño 16 columnas por 2 filas. Contiene 16 conexiones repartidas para datos, alimentación, configuración de contraste e ilumniación, etc.

Para evitar la conexión directa de todos estos pines con la placa base de la Arduino Uno, se ha decidido emplear un controlador I2C a la pantalla LCD, como se puede observar en la imagen 4.5. Se ha elegido el módulo *PCF8574*, el cual es un expansor de entradas y salidas digitales controladas por el I2C. Esto supone una reducción de las conexiones necesarias a 4, logrando así liberar entradas de la placa Arduino que podrán ser utilizadas con otros módulos. Las 4 conexiones necesarias son 2 para dar corriente al módulo, tanto para el fondo como los dígitos; Voltaje en Corriente Continua (VCC) y Tierra (GND) y 2 para datos, las cuales multiplexan las 8 inicialmente requeridas ofreciendo el mismo resultado.

Para el correcto uso de la pantalla LCD junto del controlador I2C, se ha utilizado una librería específica, LCDi2c [38], que contempla todas las adaptaciones digitales sobre el módulo y crea una serie de métodos para poder llamar desde el programa principal e interactuar con él.

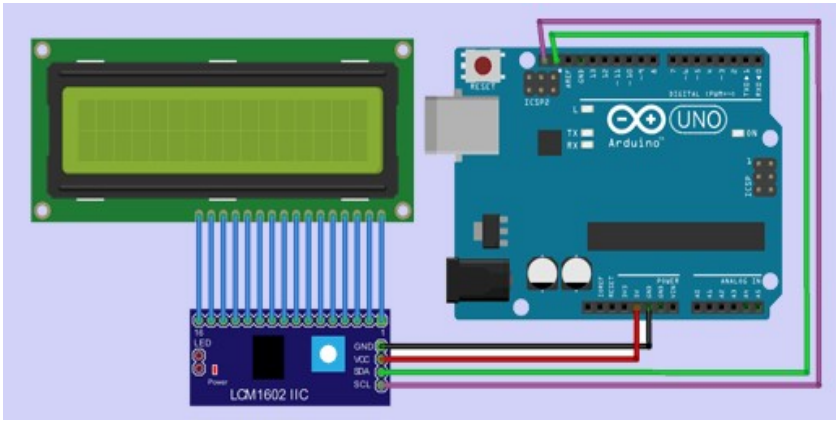


Figura 4.5: Display LCD con controlador I2C

```
1  LiquidCrystal_I2C lcd(0x27,16,2);
2  lcd.init();
3  lcd.backlight();
4  lcd.clear();
5
6  void imprimir()
7  {
8     lcd.setCursor(0,0);
9     lcd.print("Hola mundo");
10 }
```

Código 4.6: Creación del módulo y función de representación de 'Hola mundo'

Además, mediante software se puede controlar la inicialización, la exposición de las variables, la luminosidad, etc. Por parte hardware, cuenta con un potenciómetro para aumentar o reducir el contraste. Se debe ajustar correctamente este potenciómetro, para ver los caracteres que se quieren mostrar con claridad. También cuenta con otro método para la creación de caracteres personalizados, mediante la introduciendo de un mapa de bits. De esta manera, utilizando los píxeles posibles de la pantalla, se pueden representar los patrones generados como se puede observar en el código 4.7.

```
1  uint8_t check[8] = {0x0,0x1,0x3,0x16,0x1c,0x8,0x0};
2  uint8_t cross[9] = {0x0,0x0,0x11,0xA,0x4,0xA,0x11,0x0};
3  lcd.createChar(0,check);
4  lcd.createChar(10,cross);
```

Código 4.7: Creación de dos patrones propios representables

4.5. Sistema integrado

4.5.1. Hardware

En la figura 4.6 se muestra el resultado final empleado además una placa de prototipado como módulo de intermediación, en el que se han integrado todos los elementos hardware.

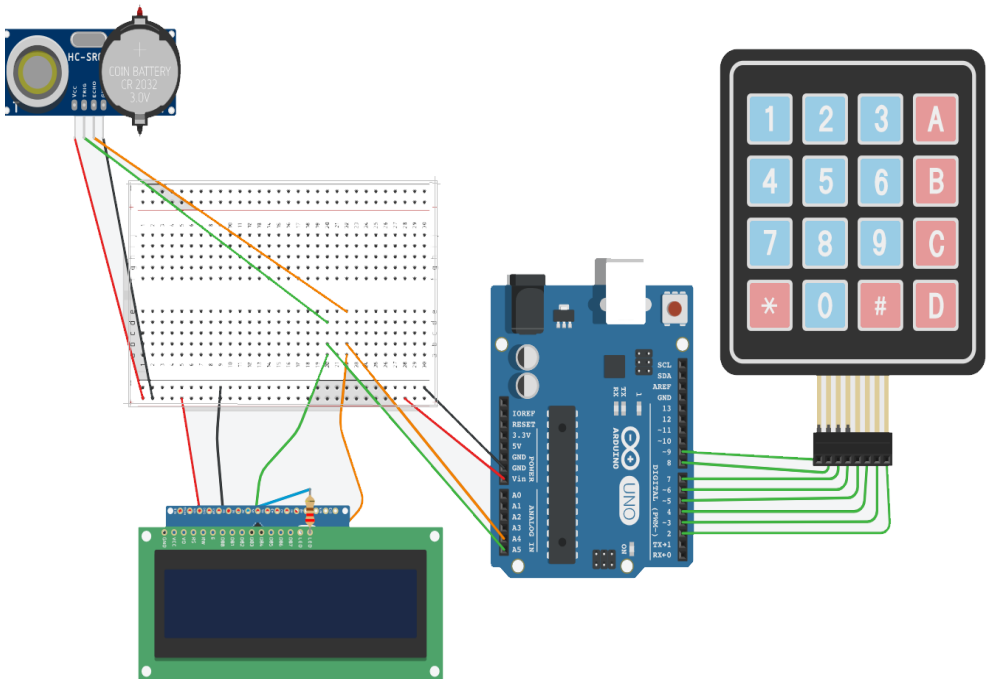


Figura 4.6: Montaje de la placa Arduino con sus módulos

En dicha figura se observan todos los módulos utilizados. En la parte de superior izquierda se observa el módulo RTC, en la parte inferior se encuentra la pantalla LCD junto con su módulo I2C y en la parte de la derecha se sitúa el teclado matricial, todo ello conectado hacia la placa Arduino Uno mediantes cables dupont.

En conjunto, los pines usados suman un total de 12. Dos son de alimentación (VCC y GND), 8 digitales (pines 2, 3, 4, 5, 6, 7 y 8) y 2 analógicos (A4-Pin de Datos (SDA) y A5-Pin de Reloj (SCL)).

El esquema implementado sobre componentes reales se puede ver reflejado en la figura 4.7.

Como se puede observar en estas dos figuras 4.6 y 4.7, tiene un reducido tamaño y consecuentemente hace que sea muy sencillo instalarlo en cualquier espacio pequeño, dejando solamente visibles las partes con las que interactuará el usuario. El consumo que tiene no es muy grande, por lo que podría alimentarse con una batería o un combinado de pilas. En su defecto, puede ir conectado a una fuente de alimentación externa de corriente continua.

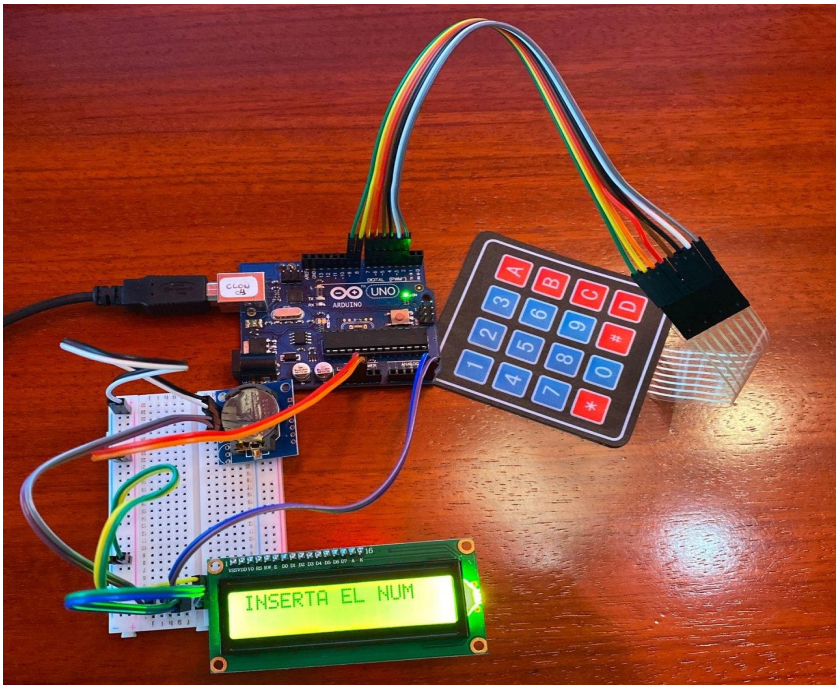


Figura 4.7: Esquema del montaje de la placa Arduino con sus módulos

4.5.2. Operativa

A continuación se describe la operativa del sistema completo. Se ha realizado una programación secuencial, separada en funciones, donde cada una realiza su propósito y finalmente se coordinan con el programa principal (*main*).

La primera interacción consiste en la recepción del número de teléfono (identificador) mediante el teclado, el cual será visible en la pantalla, para que el visitante pueda ver los dígitos introducidos. Las teclas que no se corresponden con números estarán inutilizadas por software. Cuando se pulsen estas teclas, el sistema interpretará que se ha introducido una tecla errónea y no realizará ninguna acción. Las teclas 'B' (Borrar) y 'D' (*Delete*) son la excepción y tienen un propósito específico. Mediante la tecla 'D' se permite borrar el último carácter introducido mientras que la tecla 'B' toda la secuencia introducida. Se han elegido las teclas 'B' y 'D' por su concordancia con la palabra 'borrar' en español e inglés. No obstante, estas instrucciones no van a ser mostradas en la pantalla LCD, debido a que el tamaño es muy pequeño como para mostrar oraciones legibles, por lo que se sugiere colocar un cartel junto al teclado que indique esta funcionalidad.

Así la primera interfaz con la que se va a encontrar el usuario es la pantalla principal de espera, donde se muestra la fecha y hora actuales, y el mensaje "INSERTA EL NUM", como se muestra en la figura 4.8. De esta manera, el usuario puede corroborar de manera directa si la

hora que debía acceder es la actual.

Figura 4.8: GUI en el estado inicial

Una vez que se pulse una tecla válida en el teclado matricial y se empiece a introducir el número de teléfono, la pantalla cambiará de formato. Seguirá mostrándose el mensaje de insertar el número, pero esta vez en la parte superior de la pantalla. En la parte inferior, se mostrará el número de teléfono a la vez que se va pulsando en el teclado, reflejándose en la pantalla, componiendo el número entero de 9 cifras, como se aprecia en la figura 4.9. Una vez introducido el número, la pantalla mostrará un '✓' durante 1 segundo.

Figura 4.9: GUI cuando se ha introducido el número

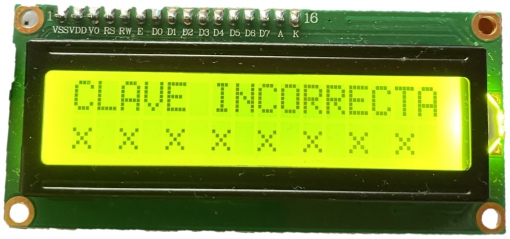
Una vez que ha pasado ese tiempo, el número ya queda almacenado y se pedirá la introducción de la clave (generada por el bot de Telegram anteriormente), donde también pueden ser borrados los caracteres introducidos erróneamente.

En la pantalla de visualización se presentará en la parte superior el mensaje ÏNSERTA LA CLAVE", dejando espacio en la parte inferior, para que se vaya mostrando la clave numérica al mismo tiempo que se va escribiendo, del mismo modo que con el número de teléfono. Esto se puede apreciar en la figura 4.10.

Figura 4.10: GUIcuando se ha insertado la clave

Introducidos el identificador y la clave, el siguiente paso es validar que la clave introducida se corresponde y permitir o no el acceso. Como ya se ha introducido para calcular la clave hace falta la hora actual. Será el módulo RTC quien supla la información temporal (fecha y hora). Una vez ya con esta información, el sistema se pone a computar para generar internamente la clave que se corresponde con los datos introducidos. Para el aprovechamiento del procesador, mientras el usuario introduce los dígitos de la clave, la placa Arduino va calculando esa clave. Una vez que el usuario aporte su clave, el sistema contrastará las dos claves, la calculada y la introducida por el usuario. En el caso de que ambas coincidan, saldrá un mensaje en la pantalla LCD, exponiendo que los datos correctos y dando acceso al recinto correspondiente. Por el contrario, si se ha accedido en una hora que no correspondía, se ha introducido un identificador erróneo o la clave aportada no es la correcta, el sistema rechazará el acceso y se notificará con un mensaje de que los datos introducidos no son correctos y por lo tanto, la clave no es correcta. Las figuras 4.11a y 4.11b muestran el resultado de este proceso.

(a) GUI clave correcta



(b) GUI clave incorrecta

Figura 4.11: Resultado del proceso de validación de clave

De esta forma, finaliza todo el proceso. Una vez salga este último mensaje, se mostrará durante 10 segundos y luego volverá a mostrarse el estado inicial hasta que se vuelva a pulsar otra tecla y esto haga que se reactive el proceso.

4.6. Estudio económico

Durante el proceso de explicación de las diferentes secciones se ha comentado que la implementación del proyecto es barata, ya que los componentes necesarios se pueden adquirir a un precio no muy elevado en cualquier tienda especializada.

La parte de implementación del bot no requiere un estudio económico concreto individual, ya que toda su implementación es de tipo software. No obstante, sí se tiene que tener en cuenta el coste económico de mantener la ejecución del programa en un servidor externo, como se había mencionado.

La cuantía económica que se ha desembolsar para realizar la implementación total del proyecto se muestra en las tablas 4.1 y 4.2. La tabla 4.1 muestra el gasto a desembolsar por parte del equipamiento hardware (Arduino), mientras que la tabla 4.2 indica el gasto de la parte software (Bot).

Material	**Cantidad**	**Coste (€)**	**Coste acumulado (€)**
Placa Arduino Uno	1	20	20
Placa Protoboard	1	2.49	22.49
Teclado Matricial	1	2.49	24.98
RTC (Módulo I2C incluido)	1	1.80	26.78
Pantalla LCD (Módulo I2C incluido)	1	5.99	32.77
Cables Dupont	40	2.49	35.26
Adaptador de corriente (AC/DC)	1	4.52	39.78

Tabla 4.1: Estudio económico de la parte hardware

Como se puede observar, el desembolso económico que se ha de hacer es muy asequible. La compra del material Arduino se puede hacer por menos de 40€. Con este dinero, se puede realizar la implementación mencionada.

El soporte mediante computación en nube al bot de Telegram se puede realizar de muchas formas. En este estudio se ha seleccionado una plataforma bastante utilizada como es Amazon Web Service (AWS). Otras opciones son Microsoft Azure, Google Cloud, Oracle Cloud, etc. Los precios se asemejan mucho entre ellos y son orientativos, ya que dependen de diversos factores como el número de interacciones, el tamaño de los datos almacenados, etc. Para este caso sobre AWS se ha realizado un presupuesto virtual en el que se cuenta con estos diferentes servicios:

- **Amazon Athenea:** Este servicio se dedica a manejar las peticiones HTTPS que se hagan al servidor y responderlas. Este servicio mide su precio a través del número total de accesos al bot diariamente y el tamaño del tráfico de datos que se produce. En este caso se va a seleccionar una media de 20 accesos diarios al bot, y con ello un traspaso de datos de 100 MegaBytes (MB). Este volumen de datos se debe a que sólo se manda información en modo texto, la cual no ocupa demasiado espacio de memoria, además es el tamaño mínimo seleccionable. Con estos datos, el coste mensual es de 26 céntimos de euro.

- **Aurora MySQL-Compatible:** Este servicio se corresponde con el alojamiento de la base de datos de forma online. Para ello se ha seleccionado la opción 'Serverless', ya que es la más económica y escalable; ideal para pequeñas bases de datos como esta, y así no incrementar su coste. Lo único que se va a contratar de este servicio es el almacenamiento, donde se han seleccionado 500 MB de memoria y hasta 1 millón de operaciones para llegar a la casilla deseada (opción mínima). Este almacenamiento solicitado es más que suficiente para manejar todos los datos necesarios. Tiene un coste de 28 céntimos de euro al mes.

- **AWS Lambda:** Este servicio es usado para la ejecución de código en el servidor, donde sólo se paga cuando se ejecute mediante una petición de un cliente. Para este caso, el servicio cuenta con la "función Lambda" gratuita, por lo que no se ha de desembolsar más dinero por este servicio.

Estos servicios, hay que añadirles el Impuesto de Valor Añadido (IVA), por lo que el total pagado por el cómputo en nube y almacenamiento de datos en la base de datos tiene un coste de 0.65€/mes.

Material	**Cantidad**	**Coste (€)/mes**
Servidor de hosting AWS	1	0.65

Tabla 4.2: Estudio económico de la parte software

Omitiendo el gasto de la mano de obra, mantenimiento, etc. el desembolso total es de 39.78€ como pago inicial y un pago mensual de 65 céntimos de euro.

5 Conclusiones y líneas futuras

Para finalizar con el Trabajo Fin de Grado, se van a exponer ciertos aspectos generales a modo de resumen. Por ello, lo primero que se va a tratar la evolución y dificultades afrontadas en el desarrollo del proyecto. Luego, se formarán unas conclusiones obtenidas al finalizar el proyecto, que dan forma a una visión general e introspectiva del resultado final obtenido, así como una mención especial a un programa que se ha usado durante todo el desarrollo de este trabajo; GitLab. Por último y a modo de cierre, se detallará el futuro que puede tener el proyecto, así como mejoras, implementaciones, desarrollos, etc.

5.1. Evolución del trabajo

Durante la realización del proyecto, aún habiendo resultado exitosa, no todo ha salido como se planteaba, sino que varias veces surgieron problemas, algunos más graves que otros, que finalmente se pudieron solventar con determinismo.

Estos, se pueden dividir en dos grupos: el primero, representando los problemas directos e implícitos del trabajo, englobando todas las partes funcionales tanto teóricas como prácticas, es decir, todo lo referido al proyecto desde un punto de vista interno, y el segundo, que incluye los problemas vistos desde un punto de vista externo, es decir, todos los inconvenientes surgidos con el ámbito de trabajo, el material, etc. Todos ellos han tenido su influencia en el trabajo, pues ha conllevado tiempo identificarlos, encontrar la forma de solventarlos y solucionarlos.

Los problemas internos han sido los mayoritarios y están referidos a la programación en sí y a la orientación que se ha llevado el proyecto. Lo primero que se empezó a hacer fue programar el bot de Telegram. La elección del lenguaje no fue fácil. Hubo bastantes inconvenientes, ya que se trataba de un lenguaje nuevo, que no se había utilizado antes. El uso de Javascript_{10} ha supuesto el estudio del mismo de forma autónoma mediante tutoriales, manuales, etc.

Otro gran problema surgido, en este caso externo, es el COVID-19. Debido a la pandemia producida por este virus junto con la cuarentena consecuente, se han producido muchas dificultades para seguir la realización normal del proyecto. El material necesario para la realización del proyecto, como son la placa Arduino y los módulos (a excepción de la placa principal), han tenido que ser provistos individualmente por terceras empresas, con retardo en las entregas o la falta de existencias en algún producto. No obstante, se ha buscado alternativas, para minimizar las alteraciones que se han podido causar.

Respecto de la solución final ofrecida, no se han realizado muchos cambios sobre el planteamiento inicial. En un inicio se había previsto la creación de 2 bots, uno para cada tipo de usuario, pero luego se vio que podía ser interesante incluir los 2 en un mismo bot, al que se le ramificasen los 2 tipos. Otra idea que se tenía en un principio era el envío de la clave por parte del bot al

visitante, pero eso requería que este tuviese activado el bot, con lo cual, no aportaría ningún beneficio. La metodología implementada permite que el visitante pueda consultar la próxima fecha y clave cuando quiera.

Un cambio que si ha resultado significativo y tiene bastante repercusión sobre todo en la implementación del dispositivo de validación, ha sido la modificación de los mecanismos de generación de las claves. En un principio se había planteado utilizar un SHA256 en vez de un SHA1. Esto fue descartado por la complejidad computacional que se necesita. La diferencia entre ellos, es que realiza muchos más pasos y al final se obtiene una ráfaga de 32 bytes en vez de 20. Esto es redundante, ya que al final, todos esos bytes se van a reducir hasta formar la clave de 6 caracteres.

5.2. Conclusiones

En este TFG se ha creado una herramienta práctica, capaz de usarse habitualmente como sustituto de las convencionales cerraduras a la hora de dar acceso a propiedades privadas a personas ajenas al lugar. Se ha demostrado que de una manera sencilla con tecnología barata en términos económicos, y sencilla a efectos prácticos de uso, se puede garantizar una seguridad bastante más alta, en comparación con la que se tiene actualmente. Las personas que se dispongan a implementarlo no se han de preocupar de instalar nuevo software en sus dispositivos más las aplicaciones de mensajería que ya suelen tener instaladas. Se ha demostrado, por un lado, la usabilidad que puede llegar a ofrecer un bot de Telegram, junto con su programación implícita, y por otro lado, la sencillez que aporta Arduino para crear un pequeño sistema de autenticación.

Se ha remarcado el uso de los lenguajes de programación usados, Node JS y Arduino, para la creación del bot y el sistema Arduino, los cuales han sido beneficiosos y aptos para este proyecto. Estos han dotado a este TFG de las características necesarias para que funcione rápida y eficaz. Sin demora notable para la apreciación humana, realiza las operaciones pertinentes y obtiene los resultados esperados.

A lo largo de todo el proyecto se ha utilizado la herramienta Gitlab para ir guardando todo el contenido generado, tanto de la parte del bot de Telegram, como de Arduino y de la memoria, guardando así el historial de versiones que se van adjuntando en el git.

5.3. Líneas futuras

Este trabajo ha servido como inicialización de un proyecto ambicioso para sustituir los mecanismos tradicionales de seguridad, por este sencillo y práctico mecanismo. Sin embargo, el proyecto actual puede ser mejorado para aportar más seguridad y fiabilidad, para abarcar más sectores o simplemente para aumentar sus funcionalidades o su comodidad de uso.

- Una primera mejora está relacionada con el tiempo indicado de entrada para cada usuario. Actualmente, los plazos son constantes de 1 hora. Pueden darse casos en los que no se quiera tener tanto tiempo una clave disponible, por lo que se podría plantear el concepto de flexibilizar ese tiempo, donde el usuario podría indicar una validez temporal más exacta,

tanto de inicio, como de fin de la efectividad de la clave. Cuando el bot le diga al usuario que introduzca la fecha y hora a la que se desea acceder, se debería añadir otro mensaje para establecer esta sentencia.

- Puesto que la lógica es desplegar la solución en más de un lugar que se quiera proteger, se debería realizar un mecanismo de gestión de varios lugares, a los cuales quieres aplicarles el mismo modelo de control de acceso. En este sentido, se debería incluir un nuevo apartado en el bot donde se requiera la introducción de un identificador único del lugar en sí (ya sea por mensajes desplegables o introducción por teclado), y que esto quede reflejado en la clave temporal final para mejorar la diferenciación con el resto de lugares adscritos.

- Otra linea de trabajo puede orientarse a habilitar conectividad para el dispositivo de validación. De esta forma se podría potenciar un traspaso más dinámico de claves donde no haría falta verificar temporalmente la clave como ocurre actualmente, sino que se podría almacenar en una base de datos compartida el identificador junto con su clave temporal asociada y simplemente comprobando esos factores debería bastar. Además, a posteriori, cuando la persona haya accedido, se puede proceder a eliminar esa clave temporal de la base de datos para evitar su reuso.

- Otro concepto que se puede ampliar es el identificador. Actualmente, el identificador para acceder se corresponde íntegramente con el número de teléfono del visitante. Además está limitado a 9 dígitos, tal y como son los números de teléfonos que se asignan a móviles en España. Se puede ampliar este rango por ejemplo para preguntar el país donde reside y dependiendo de ello se modifique la longitud de entrada del teléfono, en ambas entornos, bot y sistema validador.

- Otro mecanismo de autenticación que se puede emplear es el uso de códigos QR. Esta modificación se puede diseñar de varias maneras, dos de las cuales se describen a continuación:

  - El bot realiza la creación de un códigos de Respuesta Rápida (QR) mediante los parámetros requeridos introducidos (fecha y hora - identificador) y lo almacena en formato 'foto'. El visitante cuando requiera la clave se le concederá el código QR, donde deberá acudir al lugar donde esté implementado el sistema Arduino, el cual dispondrá de un lector de códigos QR, donde simplemente con la identificación del mismo, se detectaría que usuario y clave asociada tiene.
  - El bot realiza las operaciones para obtener la clave y lo almacena en una base de datos online. Posteriormente, cuando el visitante vaya a acceder, tendrá que escanear un código QR temporal que se le mostraría en una pantalla conectada en el sistema Arduino. En ese momento, el usuario debería capturar ese código QR y remitir al bot, el cual descifraría y validaría. Si resulta válido, el bot mandaría una señal al sistema de validación, para conceder la validez de la clave.

- Por último, concluir con una idea que ha surgido de una problemática identificada durante la realización del trabajo. El desarrollo actual no permite un trabajo concurrente y no puede atender a más de un usuario de forma simultánea. Si bien en un piloto es factible, en un producto depurado y comercial, ésto debería ser posible.

Estas son varias ideas, las cuales se han propuesto para potenciar el proyecto, pero no quiere decir que sean las únicas al tratarse de un proyecto innovador y que engloba muy diversas tecnologías.

Bibliografía

[1] Statista. *Porcentaje de hogares equipados con teléfono móvil en España de 2005 a 2019*. [En línea]. Enlace: https://es.statista.com/estadisticas/718350/porcentaje-de-viviendas-con-telefono-movil-espana/
[Consulta: 23 de marzo de 2020]

[2] SALTO Systems. *SALTO inspired access*. [En línea]. Enlace: https://www.saltosystems.com/es/
[Consulta: 10 de mayo de 2020]

[3] Tessa Assa Abloy. *TESA*. [En línea]. Enlace: https://www.tesa.es/es/site/tesa/
[Consulta: 10 de mayo de 2020]

[4] HID Global. *HID powering Trusted Identities.* [En línea]. Enlace: https://www.hidglobal.com/node/25719
[Consulta: 10 de mayo de 2020]

[5] FAC Seguridad. *FAC Seguridad*. [En línea]. Enlace: http://www.fac-seguridad.es/
[Consulta: 10 de mayo de 2020]

[6] CISA. *CISA*. [En línea]. Enlace: https://www.cisa.com/es/index.html
[Consulta: 10 de mayo de 2020]

[7] Whatsapp. *Whatsapp Messenger*. [En línea]. Enlace: https://www.whatsapp.com/?lang=es
[Consulta: 10 de mayo de 2020]

[8] Whatsapp. *Cómo comenzar a usar WhatsApp Messenger*. [En línea]. Enlace: https://www.whatsapp.com/coronavirus/get-started?lang=es
[Consulta: 10 de mayo de 2020]

[9] Whatsapp. *Whatsapp Encryptation Overview*. [En línea]. Enlace: https://scontent.whatsapp.net/v/t61.22868-34/68135620_760356657751682_6212997528851833559_n.pdf/WhatsApp-Security-Whitepaper.pdf?_nc_sid=41cc27&_nc_ohc=V1DA9CamfmMAX_KvkcY&_nc_ht=scontent.whatsapp.net&oh=8cf06903dd6aa8a89f6166ac7603ae2e&oe=5F175953
[Consulta: 10 de mayo de 2020]

[10] Whatsapp. *Información acerca de los límites de reenvío*. [En línea]. Enlace: https://faq.whatsapp.com/general/coronavirus-product-changes/about-forwarding-limits/?lang=es
[Consulta: 14 de mayo de 2020]

[11] Telegram FZ-LLC. *Telegram para Android*. [En línea]. Enlace: https://play.google.com/store/apps/details?id=org.telegram.messenger&hl=es
[Consulta: 23 de marzo de 2020]

[12] Telegram FZ-LLC. *Telegram para IOS*. [En línea]. Enlace: https://apps.apple.com/es/app/telegram-messenger/id686449807
[Consulta: 23 de marzo de 2020]

[13] Telegram FZ-LLC. *Telegram para Windows Phone*. [En línea]. Enlace: https://www.microsoft.com/es-es/p/telegram-messenger/9wzdncrdzhs0?rtc=1&activetab=pivot:overviewtab
[Consulta: 24 de marzo de 2020]

[14] Telegram FZ-LLC. *Telegram Desktop*.[En línea]. Enlace: https://desktop.telegram.org/
[Consulta: 24 de marzo de 2020]

[15] Telegram. *Telegram Web*. [En línea]. Enlace: https://web.telegram.org/
[Consulta: 24 de marzo de 2020]

[16] Telegram. *Telegram Source Code*. [En línea]. Enlace: https://telegram.org/apps#source-code
[Consulta: 24 de marzo de 2020]

[17] Mozilla. *Javascript*. [En línea]. Enlace: https://developer.mozilla.org/es/docs/Web/JavaScript
[Consulta: 30 de marzo de 2020]

[18] Mozilla. *Promise*. [En línea]. Enlace: https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promise
[Consulta: 30 de marzo de 2020]

[19] UPV. *Conceptos básicos de Programación Orientada a Objetos*. [En línea]. Enlace: http://www.upv.es/amiga/43.htm
[Consulta: 30 de marzo de 2020]

[20] OpenJS Foundation. *Node JS*. [En línea]. Enlace: https://nodejs.org/
[Consulta: 7 de abril de 2020]

[21] Arduino S.r.l. *Arduino*. [En línea]. Enlace: https://www.arduino.cc/en
[Consulta: 7 de abril de 2020]

[22] cplusplus.com. *Cplusplus*. [En línea]. Enlace: https://www.cplusplus.com/
[Consulta: 8 de abril de 2020]

[23] Red Hat. *¿Qué es el open source?*. [En línea]. Enlace: https://www.redhat.com/es/topics/open-source/what-is-open-source
[Consulta: 8 de abril de 2020]

[24] Telegram. *MTProto Mobile Protocol*. [En línea]. Enlace: https://core.telegram.org/mtproto
[Consulta: 22 de junio de 2020]

[25] Julián Pérez Porto y María Merino. *Definición de Plugin*. [En línea]. Enlace: https://definicion.de/plugin/
[Consulta: 10 de abril de 2020]

[26] Debian. *Debian, The universal operating system.* [En línea]. Enlace: https://www.debian.org/index.es.html
[Consulta: 11 de abril de 2020]

[27] Mary Bellis. *The History of the Electric Telegraph and Telegraphy.* [En línea]. Enlace: https://www.thoughtco.com/the-history-of-the-electric-telegraph-and-telegraphy-1992542
[Consulta: 7 de julio de 2020]

[28] Statista. *Conocimiento y uso de la aplicación de mensajería instantánea Whatsapp en España en 2015 y 2016.* [En línea]. Enlace: https://es.statista.com/estadisticas/509381/conocimiento-y-uso-de-la-aplicacion-de-mensajeria-instantanea-whatsapp-por-los-internautas-en-espana/
[Consulta: 7 de julio de 2020]

[29] Statista. *Ranking de las principales redes sociales a nivel mundial según el número de usuarios mensuales activos en enero de 2020.* [En línea]. Enlace: https://es.statista.com/estadisticas/600712/ranking-mundial-de-redes-sociales-por-numero-de-usuarios/

[30] Marta Juste. *WhatsApp limita el reenvío de mensajes a un solo chat cada vez para evitar bulos.* [En línea]. Enlace: https://www.expansion.com/economia-digital/companias/2020/04/07/5e8c3427e5fdea9e718b4593.html#:~:text=Compa%C3%B1%C3%ADas-,WhatsApp%20limita%20el%20reenv%C3%ADo%20de%20mensajes%20a%20un,cada%20vez%20para%20evitar%20bulos&text=La%20compa%C3%B1%C3%ADa%20ha%20anunciado%20a,a%20un%20chat%20cada%20vez.
[Consulta: 7 de julio de 2020]

[31] Arduino.cl. *Arduino UNO.* [En línea]. Enlace: https://arduino.cl/arduino-uno/
[Consulta: 12 de abril de 2020]

[32] ATMEL. *Datasheet: ATmega48/88/168.* [En línea]. Enlace: https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf
[Consulta: 12 de abril de 2020]

[33] SEGITTUR. *PDF: Sistemas de mensajería instantánea para la empresa.* [En línea]. Enlace: https://www.segittur.es/opencms/export/sites/segitur/.content/galerias/descargas/documentos/Informe-Sistema-de-Mensajera-Instantnea-para-la-empresa-ok.pdf
[Consulta: 14 de abril de 2020]

[34] Dan's Tools. *What is the unix time stamp?.* [En línea]. Enlace: https://www.unixtimestamp.com/index.php
[Consulta: 2 de mayo de 2020]

[35] Mark Stanley y Alexander Brevig. *Keypad library for Arduino.* [En línea]. Enlace: https://github.com/Chris--A/Keypad
[Consulta: 25 de mayo de 2020]

[36] Paul Stoffregen. *DS1307RTC Library*. [En línea]. Enlace: https://github.com/PaulStoffregen/DS1307RTC/
[Consulta: 25 de mayo de 2020]

[37] Paul Stoffregen. *Arduino Time Library*. [En línea]. Enlace: https://github.com/PaulStoffregen/Time/
[Consulta: 25 de mayo de 2020]

[38] Arduino. *LCDi2c Library*. [En línea]. Enlace: https://playground.arduino.cc/Code/LCDi2c/
[Consulta: 13 de mayo de 2020]

[39] Krawczyk, et. al. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. [En línea]. Enlace: https://tools.ietf.org/pdf/rfc2104.pdf
[Consulta: 4 de mayo de 2020]

[40] Eastlake 3rd & Hansen. *US Secure Hash Algorithms (SHA and HMAC-SHA)*. RFC 4634. [En línea]. Enlace: https://tools.ietf.org/pdf/rfc4634.pdf
[Consulta: 4 de mayo de 2020]

[41] Eastlake & Jones. *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174. [En línea]. Enlace: https://tools.ietf.org/pdf/rfc3174.pdf
[Consulta: 4 de mayo de 2020]

[42] Mozilla. *Hash*. [En línea]. Enlace: https://developer.mozilla.org/en-US/docs/Glossary/hash
[Consulta: 6 de mayo de 2020]

[43] Jordi Boggiano. *SHA1 + HMAC in JS*. [En línea]. Enlace: https://gist.github.com/Seldaek/1730205
[Consulta: 6 de mayo de 2020]

[44] Daniel Knuettel. *Cryptosuite2*. [En línea]. Enlace: https://github.com/daknuett/cryptosuite2
[Consulta: 6 de mayo de 2020]

[45] M'Raihi, et al. *HOTP: An HMAC-Based One-Time Password Algorithm*. RFC 4226. [En línea]. Enlace: https://tools.ietf.org/html/rfc4226
[Consulta: 7 de mayo de 2020]

[46] Jonas Hermsmeier. *HMAC-Based One-Time Password (HOTP)*. [En línea]. Enlace: https://www.npmjs.com/package/hotp
[Consulta: 8 de mayo de 2020]

[47] Arduino. *SimpleHOTP*. [En línea]. Enlace: https://www.arduinolibraries.info/libraries/simple-hotp
[Consulta: 8 de mayo de 2020]

[48] Casiano Rodríguez León. *La Función fork*. [En línea]. Enlace: https://campusvirtual.ull.es/ocw/pluginfile.php/2173/mod_resource/content/0/perlexamples/node66.html
[Consulta: 10 de junio de 2020]

[49] Maxim Integrated ™. *DS1307 64 x 8, Serial, I2C Real-Time Clock*. [En línea]. Enlace: `https://datasheets.maximintegrated.com/en/ds/DS1307.pdf`
[Consulta: 20 de junio de 2020]